Universidad Zaragoza
1542

Trabajo Fin de Grado

Estudio de la influencia del autoencoder en el problema de diagnóstico de la enfermedad de Alzheimer a partir de datos tabulares

Study of the influence of the autoencoder in the problem of Alzheimer's disease diagnosis from tabular data

Autor/es

Alejandro Facorro Loscos

Director/es

Mónica Hernández Giménez

Elvira Mayordomo Cámara

Grado en Ingeniería Informática

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2024

Índice de abreviaturas

CN	Cognitive Normal
MCI	Mild Cognitive Impairment
AD	Alzheimer's Disease
sMCI	stable Mild Cognitive Impairment
pMCI	progressive Mild Cognitive Impairment
LMCI	Late MCI
EMCI	Early MCI
ADNI	Alzheimer's Disease Neuroimaging Initiative
PET	Positron Emission Tomography
FDG-PET	Fluorodeoxyglucose PET
ApoE	Apolipoprotein E
SNPs	Single-Nucleotide Polymorphism
ADAS-Cog	Alzheimer's Disease Assessment Scale–Cognitive Subscale
MRI	Magnetic Resonance Imaging
FDG	Fluorodeoxyglucose
AV45	Florbetapir
AV1451	Flortaucipir
DTI	Diffusion Tensor Imaging
CSF	Cerebrospinal Fluid
CS	Cognitive Scores
BIO	Biomarkers
ROIs	Region of Interest
dx-bl	Diagnosis baseline
kNN	k-Nearest Neighbors

SVM	Support Vector Machine
xGB	extreme Gradient Boosting
ReLU	Rectified Linear Unit
MAE	Mean Absolute Error
CCE	Categorical Cross Entropy
BCE	Binary Cross Entropy
ROC	Receiver Operating Characteristic
AUC	Area under the ROC Curve
TPR	True Positive Rate
FPR	False Positive Rate

AGRADECIMIENTOS

Quiero expresar principalmente mi agradecimiento a las mis tutoras, Mónica Hernández Giménez y Elvira Mayordomo Cámara, ya que me han ayudado en todo lo posible y gracias a su paciencia, este Trabajo de Fin de Grado se ha podido realizar. Y que sin la sugerencia que me hicieron sobre la temática de este trabajo no habría aprendido tanto.

En segundo lugar, a mis padres, ya que sin ellos no hubiera podido cursar este grado y que en todo momento me han apoyado y me han dado ánimos para seguir estudiando. A mi hermana, que aunque no me haya dado la misma cantidad de apoyo que mis padres, lo ha hecho a su propia manera.

También quiero agradecer a mis compañeros y amigos que he hecho a lo largo de mis años en el grado, aunque hoy en día estemos cada uno en lugares diferentes, siempre recordaré aquellos momentos de hacer trabajos en la sala de estudio del Ada Byron o en la cafetería comiendo entre horas donde podíamos tener un momento de calma y risas.

Tampoco me olvido de todos los docentes que he tenido durante estos años en la Escuela de Ingeniería y Arquitectura y la huella que han ido dejando cada uno de ellos.

Por último, quiero agradecer a mi abuela, que aunque hoy en día no esté con nosotros, ella fue la razón por la cual decidí lanzarme a la realización de este Trabajo de Fin de Grado sin importar el tiempo que me llevase.

Resumen

La enfermedad de Alzheimer es la forma más común de demencia, siendo responsable de entre el 60 % y el 80 % de los casos de demencia. Según la Organización Mundial de la Salud, aproximadamente 60 millones de personas a nivel global padecen Alzheimer, siendo el 8.1 % mujeres y el 5.4 % hombres mayores de 65 años. En la actualidad, la demencia es una de las principales causas de discapacidad y dependencia entre las personas de edad en el mundo entero y la séptima causa de muerte.

Cuando se realiza el diagnóstico de un posible paciente de Alzheimer hay que seguir un proceso divido en varias etapas: la primera de ellas se basa en la detección de la enfermedad mediante la observación del paciente y los historiales médicos del mismo. Tras esto se realiza una serie de pruebas como: análisis de sangre o evaluaciones cognitivas, *Positron Emission Tomography (PET)* y *Fluorodeoxyglucose PET (FDG-PET)*, que buscan evaluar al paciente y establecer el estado de la enfermedad. La última etapa está compuesta por un análisis de los niveles de proteínas del líquido cefalorraquídeo o una detección de la proteína beta amiloide en el cerebro, es en estas pruebas donde se diagnostica el Alzheimer con mayor certeza. Este conjunto de fases ayudan al experto clínico a detectar si el paciente padece de Alzheimer pero no indican la etapa en la cual está, por lo cual es necesario la creación de nuevas medidas para una detección temprana y así poder tratar la enfermedad en fases intermedias.

Esto nos lleva a los modelos de aprendizaje, ya que estos se basan en la búsqueda de pequeños patrones o información relevante de un conjunto de datos, estos modelos se usan como modelos de ayuda del diagnóstico, por ende, con la combinación de estos modelos y el conocimiento de un médico experto se puede tener una mayor precisión y rapidez a la hora de predecir el diagnóstico del paciente.

El objetivo de este trabajo se basa en el estudio de un modelo basado en autoencoder, siendo utilizado para el problema de diagnostico de enfermedades mediante la reducción de la dimensionalidad del conjunto de datos tabulares así como modelo de apoyo a los modelos de aprendizaje tradicional en la predicción de la enfermedad de Alzheimer. Se realiza una predicción entre los casos *Cognitive Normal (CN)/ Mild Cognitive Impairment (MCI) / Alzheimer's Disease (AD)* y dentro de los pacientes diagnosticados con MCI, se predice entre los casos *stable Mild Cognitive Impairment (sMCI) / progressive Mild Cognitive Impairment (pMCI).*

Índice

Capítulo 1

Introducción y objetivos

La enfermedad de Alzheimer (*Alzheimer's disease* - AD) es una enfermedad neurodegenerativa que se produce debido al deterioro progresivo de un conjunto de estructuras neuronales. Es un trastorno neurodegenerativo incurable y terminal que lleva a sus pacientes a la pérdida de la memoria inmediata, a dificultar la capacidad de pensamiento y razonamiento e incluso a que la persona afectada por esta enfermedad tenga cambios en su personalidad y conducta. El Alzheimer es la forma más común de demencia, siendo responsable de entre un 60 % y un 80 % de los casos de la misma [1]. El factor de riesgo de desarrollar esta enfermedad aumenta con la edad, siendo así las personas mayores a 65 años las más probables a padecerlo, pero el Alzheimer no sólo es una enfermedad de la vejez ya que puede afectar a personas jóvenes de entre 40 o 50 años o incluso menos, siendo denominada como *enfermedad de Alzheimer de inicio precoz* o *Alzheimer de inicio temprano* [2].

Aunque no tenga cura o un tratamiento reversible hasta la fecha, una de las posibles soluciones es la identificación temprana de la misma mediante una serie de diagnósticos [3] o en otros casos más extremos, el uso de medicamentos previamente aprobados que pueden llegar a aliviar los síntomas de las enfermedades neurodegenerativas. Sin embargo, el uso repetitivo de los mismos puede conllevar a efectos secundarios y en ningún caso pararía la progresión del proceso neurodegenerativo [4].

Dentro el ámbito clínico en la investigación para tratar las enfermedades neurodegenerativas existen dos tipos de aproximaciones totalmente diferentes: la primera es el desarrollo de crear tratamientos que ralenticen o incluso reviertan la perdida de habilidades cognitivas o motoras [5]. Por otro lado, tenemos la investigación de crear diagnósticos que identifiquen las causas del proceso neurodegenerativo lo más temprano posible [6]. En este caso sería necesario detectar las causas del deterioro cognitivo leve, ya que este se encuentra en una etapa intermedia entre el deterioro

cognitivo asociado a la edad y la propia enfermedad de Alzheimer, así es posible realizar los tratamientos necesarios en aquellos pacientes que tengan una alta probabilidad de tener una progresión al Alzheimer [7] y poder ralentizar o revertir el avance de la enfermedad. Por desgracia, el desarrollo de esta segunda aproximación se ha visto negativamente afectado por el limitado conocimiento que existe en la actualidad sobre las causas y mecanismos que producen la muerte de las neuronas durante las enfermedades neurodegenerativas.

Para que se realice un diagnóstico los expertos clínicos necesitan interpretar una gran cantidad de datos de tipología variada como pueden ser: historial clínico del paciente (demografía, hábitos, raza, educación...etc), resultados de exámenes cognitivos, posibles mutaciones dentro de la información genética, escáneres cerebrales de diferentes tipos de imágenes y muchos más [8]. Este gran y diferente conjunto de datos conlleva a que la identificación de la enfermedad puede llegar a ser altamente subjetiva o incluso un desafío el diagnosticarla, ya que los expertos médicos no son capaces de poder analizar manualmente todos estos datos con rapidez o no tienen la capacidad de reconocer e identificar patrones más complejos, ya que es necesario la comparación de todas las variables del paciente con otros casos similares [9]. También este tipo de diagnósticos llega a mostrar pruebas de que existe un proceso neurodegenerativo en el paciente, pero no especifican la etapa en la que se encuentra o incluso pueden llegar a fallar, siendo un ejemplo la enfermedad de Alzheimer, donde los análisis *post mortem* han llegado a mostrar un error en los diagnósticos de hasta el 20 %. Una posible solución a este problema es el uso de sistemas computacionales para facilitar el proceso de identificación/predicción del paciente y más hoy en día debido a la explosión de los métodos *deep learning* y el incremento de relevancia y uso que está teniendo la Inteligencia Artificial en todos los sectores.

Un ejemplo del buen resultado que muestran este tipo de modelos se puede ver en la clasificación de los pacientes en varias fases de la enfermedad de Alzheimer, obteniéndose así buenos resultados con el uso de información clínica y anatómica y reconocimiento de la proteína *Apolipoprotein E (ApoE)*. Otro posible ejemplo es el del modelo *Spasov-18* [10], que obtuvo una exactitud media del 99 % identificando pacientes con Alzheimer que se encontraban dentro del grupo de pacientes CN usando el conjunto de datos *Alzheimer's Disease Neuroimaging Initiative (ADNI)*. También el modelo *Sohn-19* ha demostrado unos buenos resultados en la identificación de pacientes con deterioro cognitivo leve estable o deterioro cognitivo leve progresivo, los cuales han padecido Alzheimer en el transcurso de 6 años [11].

En resumen, los modelos *deep learning* han demostrado tener potencial como apoyo a la toma de decisiones en una variedad de enfermedades como: enfermedades del corazón [12], detectar enfermedades de los pulmones [13] o como se ha citado anteriormente para detectar el Alzheimer mediante el uso de diferentes tipos de datos [14]. Existen varias iniciativas hoy en día basadas en *multi-task learning* o *transfer learning* que se encuentran en proceso de desarrollo temprano, aunque esto no quiere decir que el desarrollo de modelos *deep learning* sea la única opción a tener en cuenta, los modelos de aprendizaje no supervisado también ayudan en la identificación de enfermedades o también aportan facilidades en la investigación las causas de las mismas. Por todo ello se desarrollan distintas soluciones para un mismo problema con el objetivo de conseguir un método *machine learning* que tenga un balance entre los datos a utilizar, la complejidad del sistema y la interpretabilidad de los resultados.

Este trabajo se ha basado en el artículo *"Multimodal deep learning models for early detection of Alzheimer's disease stage"* [15] realizado por *Janani Venugopalan et al*, dicho artículo se basa en la creación de tres modelos *deep learning* donde cada uno es entrenado con un tipo de dato diferente: imágenes, historial clínico y *Single-Nucleotide Polymorphism (SNPs)*, los cuales pueden o no estar en un paciente, es decir, el paciente puede estar registrado en los tres tipos de datos o sólo en uno. Estos modelos se encargan de reducir la dimensionalidad de cada uno de los diferentes conjuntos de datos para después realizar una unión e integración de los distintos tipos de datos a modelos de aprendizaje tradicionales para que realicen la predicción de la etapa de Alzheimer. El objetivo es demostrar que los modelos *deep learning* tienen mayor efectividad sobre los modelos tradicionales y que un modelo *deep learning multimodal* es mucho mejor que los modelos de aprendizaje tradicional. En nuestro caso se ha intentado reproducir el estudio del artículo a un solo conjunto de datos, usando un modelo *autoencoder* que reduce la dimensionalidad de la entrada de datos para que un modelo de aprendizaje tradicional prediga el diagnóstico del paciente. Se ha elegido este artículo por el interés del comportamiento de este tipo de modelos sobre datos tabulares y además por la doble meta de predecir el Alzheimer y las etapas en las cual se encuentra el paciente.

Capítulo 2

Conjunto de datos: Descripción y Preprocesado

2.1. Conjunto de datos: TADPOLE Challenge

El conjunto de datos utilizado en este proyecto provienen de la competición *TADPOLE Challenge* [16]. El objetivo de esta competición consiste en la creación de un modelo que muestre los mejores resultados en base a una predicción mes a mes de tres variables diferentes de cada uno de los pacientes: diagnóstico, volumen ventricular y puntuación de *Alzheimer's Disease Assessment Scale–Cognitive Subscale (ADAS-Cog)*. Para ello se les da a los participantes varios conjunto de datos que se encuentran en la base de datos ADNI. Los conjuntos de datos se dividen en *datasets* (D1 a D4), siendo el conjunto de datos D1 utilizado para entrenar el modelo, el conjunto de datos D2 y D3 utilizado para realizar una predicción de los datos, y por último, el conjunto de datos D4 sirve para testear el modelo. En este proyecto nos centraremos en usar el *dataset* D1-2 ya que es el que contiene la mayor cantidad de información, siendo ideal para entrenar nuestra red neuronal. Para la evaluación de nuestro modelo utilizaremos los pacientes del *dataset* D4 que se encuentren en el *dataset* D1-2. Para evitar *data leakeage*, borraremos aquellos pacientes del conjunto D1-D2 que se encuentren en D4.

Dentro del *dataset* podemos encontrar diferentes tipos de datos: pruebas cognitivas, *Magnetic Resonance Imaging (MRI)*, tomografías por emisión de positrones de tres tipos: *Fluorodeoxyglucose (FDG)*, *Florbetapir (AV45)*, *Flortaucipir (AV1451)*, *Diffusion Tensor Imaging (DTI)*, *Cerebrospinal Fluid (CSF) biomarkers*, información de la apolipoproteína E (ApoE), siendo esta un factor de riesgo para el desarrollo de contraer Alzheimer, información demográfica del paciente (género, edad, raza ...etc), y por último, tanto el diagnóstico del paciente en esa misma consulta como el cambio de diagnóstico que ha tenido de una consulta a otra.

Nombre de las tablas	Descripción de los datos
ADNIMERGE	Información demográfica y test neuropsicológicos realizados por un experto clínico
UCSFFSL	Imágenes MRI de *Region of Interest (ROIs)* obtenidas a partir del programa FreeSurfer
UCSFFSX	
BAIPETNMRC	Biomarcadores procedentes de *Banner Alzheimer's Institute PET NMRC*
DTIROI	Imágenes DTI para entender donde están los vasos sanguíneos del cerebro, su anchura y orientación
UCBERKELEY	Biomarcadores obtenidos a partir de los análisis proporcionados por la universidad de Barkeley, California realizados a través de pruebas PET ROIs AV1451 y AV45
UPENBIOMK9	Datos de las proteínas beta-amiloide, tau y p-tau obtenidas a través de pruebas del líquido cefalorraquídeo

Tabla 2.1: Nombre y descripción de las tablas que forman *TADPOLE challenge*.

En la Tabla 2.1 se encuentran los nombres de las tablas de las diferentes bases de datos que conforman el dataset de *TADPOLE Challenge* con la descripción del tipo de dato que contiene cada una.

En el siguiente enlace de *github* se puede encontrar el resumen del *dataset TADPOLE*: `https://github.com/swhustla/pycon2017-alzheimers-hack/blob/master/docs/data_dictionary.md`, donde se puede ver los atributos que lo forman, acompañados de una descripción y el nombre de la tabla de donde provienen.

El atributo *diagnostic* será utilizado como variable dependiente para el problema CN/MCI/AD. Esta variable se ha creado a partir de la columna *DXCHANGE* que indica el cambio de diagnostico que ha tenido el paciente entre consultas, puede ser un cambio que varie entre CN, MCI ó AD y pueden ocurrir tanto reversiones como conversiones entre diagnósticos.

Se ha decidido crear esa nueva variable debido a que dentro del conjunto de datos los pacientes tienen múltiples visitas a lo largo del tiempo y su diagnóstico puede variar de una visita a otra, como este proyecto se centra más en el riesgo de predecir Alzheimer que en la progresión de la enfermedad, se ha realizado un mapeo de los datos según las reglas definidas en la Tabla 2.2.

Valores originales	Estable: CN a CN = 1 Reversión: MCI a CN = 7 Reversión: AD a CN = 9	Conversión: CN a MCI = 4 Estable: MCI a MCI = 2 Reversión: AD a MCI = 8	Conversión: CN a AD = 3 Conversión: MCI a AD = 5 Estable: AD a AD = 6
Valores después del mapeo	CN = 1	MCI = 2	AD = 3

Tabla 2.2: Reglas para el ajuste de los distintos valores de la columna *DXChange* y la obtención de la columna *diagnosis*.

Para los pacientes estables no se realiza ningún cambio en el diagnóstico mientras que para aquellos que tienen una conversión o reversión de una etapa a otra se les asigna el valor de la etapa a la cual realizan el cambio correspondiente.

En la Tabla 2.3 se muestra la cantidad de pacientes con el valor del nuevo diagnóstico tras la realización del mapeo en los *dataset* de entrenamiento y evaluación, obteniendo así tres clases diferentes: cognitivo normal (CN), deterioro cognitivo leve (MCI) o enfermedad del Alzheimer (AD).

Clases	Train (D1 - D2)	Eval (D4)
CN	3111	976
MCI	4611	968
AD	2991	69

Tabla 2.3: Número de pacientes en cada uno de los tres diferentes diagnósticos.

También se ha decidido abarcar el problema sMCI / pMCI, para ello solo se han tenido en cuenta aquellos pacientes que tengan en la columna *Diagnosis baseline (dx-bl)* el valor *Late MCI (LMCI)* o *Early MCI (EMCI)*, esto significa que tendremos en cuenta aquellos pacientes que su diagnóstico ha sido deterioro cognitivo leve en alguna de sus visitas, después dividiremos esos pacientes en dos grupos: sMCI y pMCI.

Los pacientes dentro del grupo sMCI serán aquellos pacientes que se les haya diagnosticado deterioro cognitivo leve en todos y cada uno de sus diagnósticos, mientras que los pacientes pMCI serán aquellos pacientes que tuvieron una progresión de MCI a AD en alguno de sus diagnósticos. Ambos criterios tienen en cuenta un periodo de tiempo de 3 años desde su primera visita.

En la Tabla 2.4 se puede ver la cantidad de pacientes con deterioro cognitivo leve estable y deterioro cognitivo leve progresivo tras aplicar esta serie de normas.

Clases	Train (D1 - D2)	Eval (D4)
sMCI	2833	587
pMCI	1329	66

Tabla 2.4: Número de pacientes con el diagnóstico sMCI/pMCI.

2.2. Preprocesado de los datos

Si se quiere tener una mejor calidad en los datos y que el conjunto de datos esté preparado para la etapa de entrenamiento y evaluación se necesita realizar una modificación y un control de calidad para adaptarlos al problema que queremos investigar. Para ello se han realizado los siguientes pasos:

– **Normalización y transformación de los datos:** debido a que el conjunto de datos de *TADPOLE* contiene una gran variedad de tipos cuantitativos como categóricos, siendo los datos cuantitativos aquellos que tienden a tener un gran rango de valor, es necesario realizar una normalización y transformación de los mismos, creando así un conjunto de datos estructurado y coherente para que de esta manera nuestro modelo no tenga ningún problema en la etapa de entrenamiento. Con respecto a los datos categóricos, los transformamos en datos binarios mediante el proceso *one hot encoding* , es decir, creamos una columna de valores para cada valor único que tenga el atributo de tal manera que se marca con un 1 la columna correspondiente al valor presente en cada registro dejando las demás columnas con un valor de 0. En cuanto a los datos numéricos, se normalizarán entre 0 y 1, para ello utilizaremos una normalización de los datos según la Ecuación 2.1 donde se tiene en cuenta el máximo y mínimo valor de la columna de datos.

$$z_{ij} = \frac{x_{ij} - x_{jmin}}{x_{jmax} - x_{jmin}}, \tag{2.1}$$

Donde z_{ij} es el nuevo valor del i-ésimo del feature j, x_{ij} es el valor i-ésimo de la columna correspondiente al feature j, x_{jmin} es el valor mínimo del feature y por último, x_{jmax} es el valor máximo del feature. En caso de que exista algún valor negativo dentro de la columna feature se realiza la Ecuación 2.2.

$$z_{ij} = x_{ij} + |x_{jmin}|, \tag{2.2}$$

De esta manera transformamos el valor mínimo que pasará a ser un 0 y los demás valores serán todos positivos. Hay que tener en cuenta que existen casos en los que ya hay valores positivos dentro de los negativos por lo que realizar el

valor absoluto de los valores negativos no nos ayudaría a que los datos fueran coherentes.

- **Tratamiento de valores ausentes:** en el conjunto de datos *TADPOLE* existen varios elementos que no tienen ningún valor asociado o el valor asociado es un -4, por lo que se descartarán aquellos atributos a los que les falten más del 30 % de los datos. Con respecto a aquellos atributos que no hayan sido eliminados pero que contengan valores vacíos, se les aplicará el algoritmo *k-Nearest Neighbors (kNN)* para sustituirlos por valores próximos, para ello se utilizará la clase *KNNImputer* usando un número máximo de vecinos k = 10.
- **Eliminación de datos no necesarios:** algunos atributos contienen datos del tipo fecha, tiempo o incluso identificadores de las distintas tablas, siendo así tipos de datos no relevantes en nuestro proyecto y que incluso pueden afectar negativamente al entrenamiento del mismo. Por lo que estos datos no se tendrán en cuenta en el preprocesado. Algunas columnas que se han eliminado son las siguientes: SITE, COLPROT, ORIGPROT, EXAMDATE, RUNDATE, VERSION, LONISID, LONIUID, STATUS, entre otras.

Capítulo 3

Modelos y métodos

Debido a la naturaleza de los datos tabulares, el problema pasa a tener una entrada altamente heterogénea que podría perjudicar las prestaciones de los algoritmos, esto ocurre en las redes neuronales profundas donde se generan unas prestaciones peores contra todo pronóstico, siendo superadas por el modelo *extreme Gradient Boosting (xGB)* [17]. Tomando como inspiración el artículo de referencia de *Janani Venugopalan et al* [15], se quiere investigar si un modelo *autoencoder* podría transformar los datos tabulares en un conjunto de datos más homogéneo donde estuviesen codificados los atributos más representativos del *dataset*, y si este cambio generaría una mejora en las prestaciones de los algoritmos.

El modelo que se usará como entrada de datos es un *autoencoder*. Debido a que la información con la que se está trabajando está formada por listas, donde cada valor representa un atributo del paciente que puede estar o no relacionado con los valores adyacentes, se ha desestimado el uso de redes neuronales convolucionales y se utilizarán a su vez capas densas o también llamadas *fully conected layer*.

El siguiente elemento es un modelo de aprendizaje tradicional, es decir, un modelo basado en redes neuronales con una arquitectura que contiene pocas capas ocultas. Este modelo nos ayudará a clasificar nuestros datos una vez se reduzca su dimensionalidad, ya que el *autoencoder* no tiene la capacidad de predecir las clases correspondientes. Como existen muchos modelos de aprendizaje tradicional se ha decidido estudiar el comportamiento de un conjunto específico de ellos, estos modelos se han obtenido a partir de la librería *Scikit-learn*.

Para obtener los datos a partir de ficheros *.csv* y manipular y crear *dataframes* en *python* se ha utilizado la librería *Pandas* en su última versión. Se ha usado la librería *Keras* para el entrenamiento y creación del *autoencoder*.

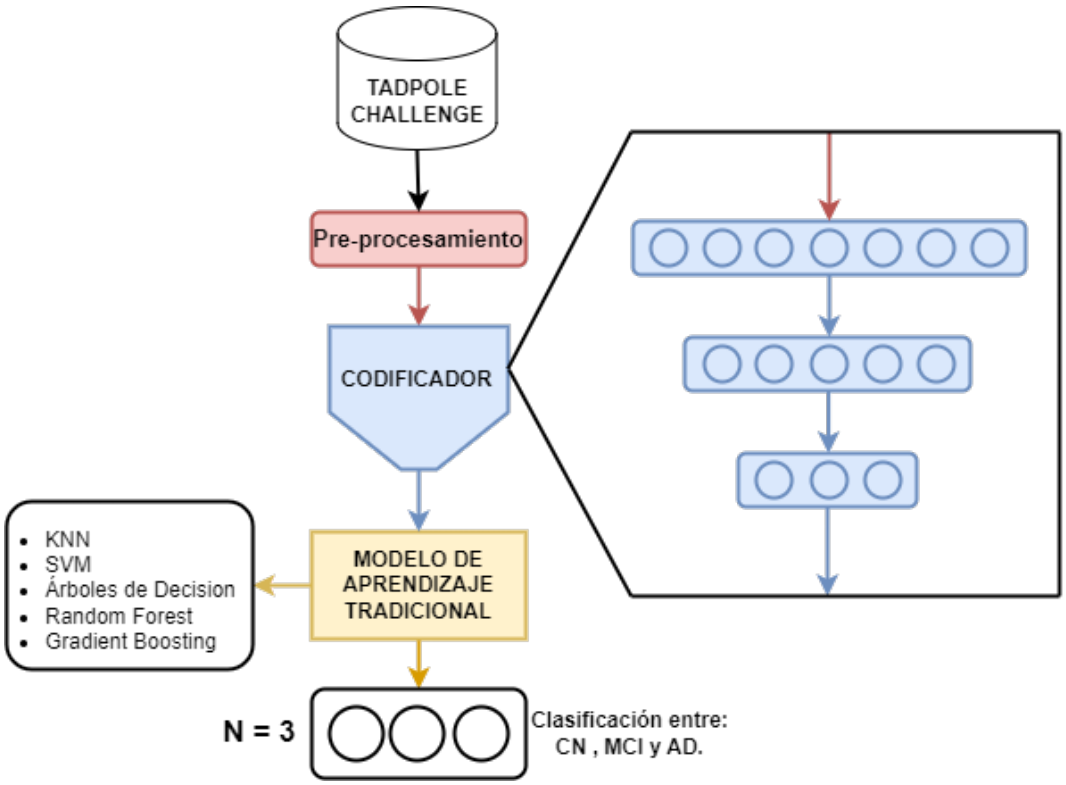


Figura 3.1: Diseño final con etapa de pre-procesamiento, codificador y modelo de aprendizaje tradicional.

En la imagen de la Figura 3.1 se puede ver el diseño del modelo que se ha implementado en este trabajo.

3.1. Autoencoder

El primer elemento de nuestro modelo es el *autoencoder*, que consiste en una serie de capas formadas por neuronas que están conectadas completamente unas con otras. El *autoencoder* que se ha creado en este trabajo es una arquitectura formada por *feed-forward layers*, esto quiere decir que la información que viaja de una capa a otra sólo toma una única dirección. También es un modelo de aprendizaje no supervisado, ya que el objetivo de este es aprender a obtener una representación codificada de los datos de entrada, por lo que le permite detectar características relevantes de los datos.

Un *autoencoder* está formado por tres elementos: el codificador o *encoder*, que reduce la dimensión de los datos de entrada, el decodificador o *decoder*, que reconstruye los datos suministrados por el codificador hasta obtener la representación original, y por último el *bottleneck*, siendo una capa central que separa *encoder* y *decoder*, y representa la dimensionalidad más pequeña del modelo.

Los datos de entrada pasan por cada una de las capas del *encoder* que se encarga de

transformar la entrada en una de mayor abstracción según la Ecuación 3.1,

$$y = f(Wx + b) \tag{3.1}$$

donde f se define como la función de activación y $[W, b]$ como los parámetros a entrenar. Tras esto se introducen los datos mapeados (y) a través de las capas del *decoder* para obtener una representación de la entrada original (x), el valor de los datos reconstruidos $\hat{x}$ se obtienen según la Ecuación 3.2.

$$\hat{x} = f(W^T y + b') \tag{3.2}$$

Donde sólo b' necesita ser entrenado ya que los pesos W^T del *decoder* están ligados a los pesos del *encoder*, es decir, son los mismos pesos pero transpuestos. Esto se consigue gracias a que durante la etapa del entrenamiento del modelo se realiza una optimización para obtener la matriz inversa de pesos W, siendo el método con menor coste computacional el de obtener una matriz transpuesta W^T, que se cumple si W es una matriz compuesta por filas ortonormales [18].

El autoencoder se ha diseñado del siguiente modo:

- Entrada: capa *input* de igual dimensión que la dimensión de atributos del conjunto de datos.
- Capa *dense* con 600 neuronas y función de activación *ReLU*.
- Capa *dense* con 300 neuronas y función de activación *ReLU*.
- Capa *dense* con 200 neuronas y función de activación *ReLU*.
- Capa *dense* con 150 neuronas, función de activación *ReLU*. Regularización *L2* del 0.01 e inicialización de los pesos *He Normal*. Capa utilizada como *bottleneck*.
- Capa *dense* con 200 neuronas y función de activación *ReLU*.
- Capa *dense* con 300 neuronas y función de activación *ReLU*.
- Capa *dense* con 600 neuronas y función de activación *ReLU*.
- Capa *dense* con un número de neuronas igual a la dimensión de atributos del conjunto de datos, función de activación *Sigmoid*.

La función de activación utilizada en el autoencoder es una función *Rectified Linear Unit (ReLU)*, ya que es la más utilizada en este tipo de sistemas de aprendizaje. Para evitar un sobre-ajuste del modelo se ha decidido añadir una capa previa *dropout* y una capa posterior *batch normalization* a cada una de las capas que componen tanto

el *encoder* como el *decoder*. La función de pérdida utilizada es *Mean Absolute Error (MAE)*, por último, se ha usado *Adam* [19] como algoritmo de optimización.

Tras entrenar las capas del autoencoder se realiza un proceso de ajuste fino (*fine-tuning*), que consiste en eliminar la capa *decoder* para añadir una capa *soft-max* que predice la clase final de los datos. La función de pérdida utilizada en este proceso es la función *Categorical Cross Entropy (CCE)*, *Adam* se mantiene como algoritmo de optimización.

Los hiperparámetros del modelo (tasa de aprendizaje, *dropout*, *batch size* ...etc) han sido optimizados utilizando *tenfold cross-validation* utilizando el 10 % del conjunto de datos de entrenamiento como conjuntos de datos de validación, el 90 % restante se mantiene como datos de entrenamiento.

3.2. k vecinos más próximos (*k-Nearest Neighbor*)

El algoritmo k vecinos más próximos se basa en seleccionar un valor de nuestro conjunto de datos, estudiando la distancia que obtiene con los demás valores restantes del conjunto, de tal manera que se quedará con un conjunto igual a k vecinos más cercanos al valor estudiado, y se realizará una votación por la clase más frecuente entre esos k vecinos que se han tenido en cuenta anteriormente. Esto se realiza para cada uno de los x valores de nuestro conjunto [20]. El número de vecinos k asignado al modelo ha sido equivalente a 3.

3.3. Máquina de Vector Soporte (*Support Vector Machine*)

El modelo *Support Vector Machine (SVM)* funciona del siguiente modo: dado un conjunto de puntos que es un subconjunto del espacio, en el que cada punto pertenece a una de las posibles n categorías, el objetivo es definir un hiperplano o varios hiperplanos que separen de forma óptima a los puntos pertenecientes a las diferentes categorías del conjunto [21]. Se ha aplicado como parámetro un *kernel* lineal.

3.4. Árboles de decisión (*Decision Tree Classifier*)

Los árboles de decisión se caracterizan por tener una estructura formada por ramas y nodos: las ramas representan la decisión en función de una determinada condición y los nodos se dividen en dos tipos, internos, que son las características que se consideran a la hora de tomar una decisión, y los nodos finales, que representan el resultado de

la decisión que en nuestro caso es la clase que corresponde al diagnóstico [22]. Se han utilizado los parámetros por defecto según *Scikit-learn*.

3.5. Bosque aleatorio (*Random Forest Classifier*)

Este modelo es el resultado de la combinación de varios árboles de decisión. Este método combina tanto el *bagging* como la selección aleatoria de atributos. El *bagging* consiste en la asociación de distintos datos de entrenamiento de tal manera que ninguno de los árboles entrenan con todos los datos de entrenamiento, y cada uno de los árboles de decisión ven un conjunto de datos diferente. Tras entrenar y obtener la decisión de cada uno de los árboles de decisión se realiza una votación por mayoría para obtener la clase final [23]. Se han utilizado los parámetros por defecto según *Scikit-learn*.

3.6. Potenciación del gradiente (*Gradient Boosting*)

Por último, se ha incluido un clasificador basado en *gradient boosting*. Dicho algoritmo se basa en mejorar el modelo base a partir de la creación de varios clasificadores más débiles, aumentando el peso de cada observación que resulta complicado clasificar. Al final se realiza una de combinación de cada uno de los clasificadores débiles creando un único clasificador [24]. Se ha utilizado un número de estimadores equivalente a 20 y un *learning rate* equivalente a 0.3.

Capítulo 4

Evaluación y resultados

4.1. Resultados Random Forest

En primer lugar se ha comprobado si las etapas realizadas durante el procesado para el conjunto de datos permiten obtener un modelo preciso. Para ello se ha utilizado como referencia el artículo de *Shaker El-Sappagh 'A multilayer multimodal detection and prediction model based on explainable artificial intelligence for Alzheimer's disease'* [25] donde se ha tomado como base los valores de precisión y *F1 Score* de las primeras tablas. En este artículo se utiliza un modelo *random forests* con parámetros por defecto junto al algoritmo *K-fold cross validation* sobre cada una de las distintas modalidades de los datos. Debido a que el conjunto de datos usado en el artículo no es el mismo que se ha utilizado en este proyecto, solo se han tenido en cuenta las siguientes modalidades de datos: *Cognitive Scores (CS)*, imágenes por resonancia magnética (MRI), tomografías por emisión de positrones (PET), tensor de difusión (DTI) y *Biomarkers (BIO)*. Además de evaluar cada una de las modalidades por separado también se ha evaluado las modalidades en conjunto.

No se puede realizar una comparación de datos con la modalidad DTI debido a que esta no se encuentra en el artículo, pero se ha evaluado para observar los valores que genera. Los atributos relacionados con el diagnóstico no se han tenido en cuenta aunque se encuentren dentro de la modalidad CS, ya que estos dependen de la variable dependiente e influyen en gran magnitud en los resultados.

En la tabla de la Figura 4.1 se puede observar la comparación entre los resultados obtenidos y los resultados del artículo.

	Artículo referencia		Datos TADPOLE	
Modalidades	**Precision**	**F1 Score**	**Precision**	**F1 Score**
Todas	**93,42 ± 2,73**	**93,39 ± 2,19**	90,80 ± 1,70	91,00 ± 1,60
CS	92,00 ± 2,26	92,08 ± 2,00	**93,10 ± 2,80**	**93,30 ± 2,70**
MRI	46,99 ± 4,01	46,50 ± 3,91	**76,80 ± 2,40**	**77,10 ± 2,40**
PET	65,23 ± 2,98	65,89 ± 2,11	**66,70 ± 2,60**	**67,50 ± 2,60**
DTI	-	-	67,30 ± 3,30	68,10 ± 3,40
BIO	58,75 ± 3,11	58,31 ± 2,89	**66,90 ± 3,00**	**67,60 ± 3,00**
CS + MRI	92,05 ± 3,99	92,06 ± 4,01	**92,10 ± 2,00**	**92,30 ± 2,00**
CS + PET	92,05 ± 3,33	92,05 ± 3,41	**93,00 ± 2,80**	**93,20 ± 2,70**
CS + DTI	-	-	93,30 ± 2,80	93,20 ± 2,70
CS + BIO	91,73 ± 2,91	91,74 ± 3,08	**93,10 ± 3,00**	**93,20 ± 2,90**
CS + BIO + MRI	**93,74 ± 4,00**	**93,72 ± 4,44**	92,10 ± 3,00	93,20 ± 2,10
CS + BIO + PET	92,89 ± 2,99	92,84 ± 3,20	**93,10 ± 3,00**	**93,20 ± 2,90**
CS + BIO + DTI	-	-	93,10 ± 2,90	93,40 ± 2,80
CS + BIO + PET + MRI	-	-	87,10 ± 1,80	87,10 ± 1,80
CS + BIO + PET + DTI	-	-	92,30 ± 2,20	92,40 ± 2,20

Tabla 4.1: Precisión y *F1 Score* de las distintas modalidades de datos.

La precisión y medida F1 mejora en algunas modalidades como CS, BIO, PET o MRI mientras que en otras modalidades son levemente inferiores en comparación a las métricas del artículo como puede ser en el caso de CS + BIO + MRI, y en la combinación de todas las modalidades. Según los resultados obtenidos en la Tabla 4.1, se han obtenido métricas similares a las métricas del artículo de referencia a pesar de que la etapas de procesado y el conjunto de datos utilizados entre proyectos difieren unos de otros.

4.2. Entrenamiento y evaluación del modelo autoencoder

4.2.1. Optimización de hiper-parámetros

A la hora de buscar una optimización en los hiper-parámetros se tuvo en cuenta el uso de las estrategias *grid search* y *random search*, pero ambas se desecharon, ya que la primera estudia cada una de las posibles combinaciones del conjunto de hiper-parámetros definidos y debido a esto, el coste computacional de nuestra optimización sería muy elevado al tener en cuenta un elevado número de variables, y respecto a *random search*,

aunque sea una evaluación aleatoria de valores dentro del conjunto de hiper-parámetros no se quiere perder una posible combinación de hiper-parámetros óptima al realizarse el algoritmo. Por lo que se decidió utilizar el algoritmo de optimización bayesiana, dado que este algoritmo arregla los problemas que generan los dos algoritmos anteriormente nombrados.

Optimización Bayesiana

La optimización bayesiana, a diferencia de la búsqueda aleatoria o búsqueda en cuadrícula, tiene en cuenta el resultado de las evaluaciones anteriores, donde se aplica una función probabilística para seleccionar la combinación que mejor resultado puede dar. Nuestro modelo pasa a ser un modelo probabilístico definido dentro de una caja negra y el valor de la función objetivo es la métrica de validación del modelo, gracias a esto, la búsqueda se va dirigiendo a regiones con mayor interés por cada iteración realizada por lo cual la cantidad de combinaciones e hiper-parámetros se reduce.

La función que se ha utilizado para esta optimización es *BayesSearchCV* aplicando el algoritmo *K-fold cross validation* definiendo como parámetro K = 10, por lo que el conjunto de datos pasa a ser un 90 % como datos de entrenamiento y el 10 % restante como datos de test. También se ha utilizado la función *early stopping* proporcionada por la librería *Keras*, con un valor de paciencia equivalente a 10 y un valor mínimo delta del 0.01 para así poder establecer un número fijo de *epochs*, por lo que para esta etapa de optimización como para futuras etapas del proyecto se utilizará esta función estableciendo un número de *epochs* equivalente a 50. En nuestra optimización se ha establecido MAE como función de pérdida y *Adam* como optimizador.

En la Tabla 4.2 se encuentran el espacio de valores establecidos para la etapa de optimización:

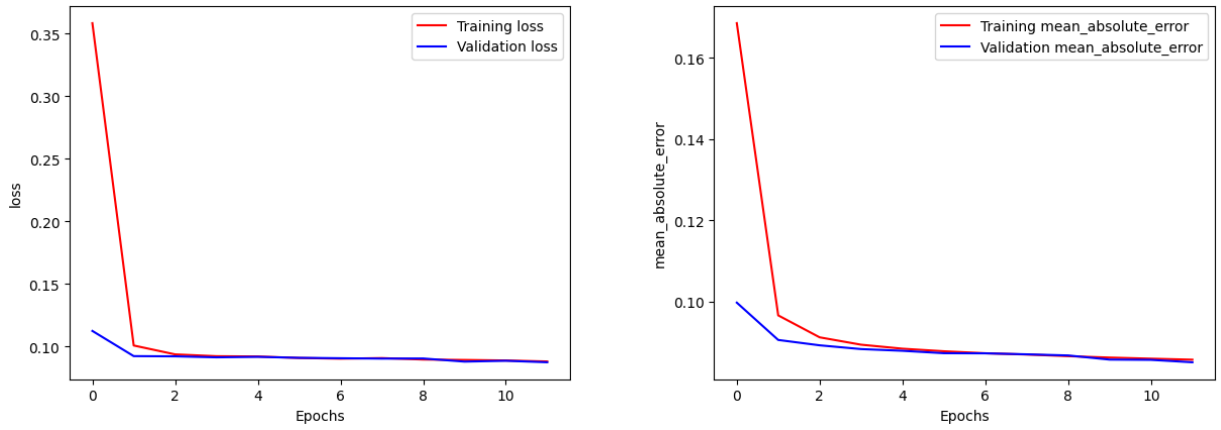
Hiper-parámetros	Espacio de valores
Learning Rate	distribución uniforme entre 1e-3 y 0.1
Batch Size	16, 32, 64
Dropout Value	0.2, 0.3, 0.4

Tabla 4.2: Espacio de valores de los hiper-parámetros usados en la optimización bayesiana.

Según la optimización bayesiana, la mejor combinación de hiper-parámetros es: 0.004 como valor de *learning rate*, 0.2 como valor *dropout* y un *batch size* igual a 32. Obteniendo un valor *mean absolute error* equivalente a 0.133.

4.2.2. Entrenamiento y evaluación del modelo autoencoder

Para la etapa de entrenamiento y evaluación del modelo se ha utilizado el algoritmo *Adam* como optimizador de modelos ya que es un algoritmo muy usado en redes neuronales debido a su capacidad de ajustar el *learning rate* según los valores del gradiente obtenidos. Como función de pérdida se ha utilizado MAE, ya que el problema se basa en generar una salida equitativa al conjunto de entrada y es una función de pérdida robusta si el conjunto de datos contiene valores atípicos o *outliers* como es en el caso del conjunto de datos *TADPOLE*. En la Figura 4.1 se muestra el valor de la función de pérdida generado por cada *epoch* en la etapa de entrenamiento.



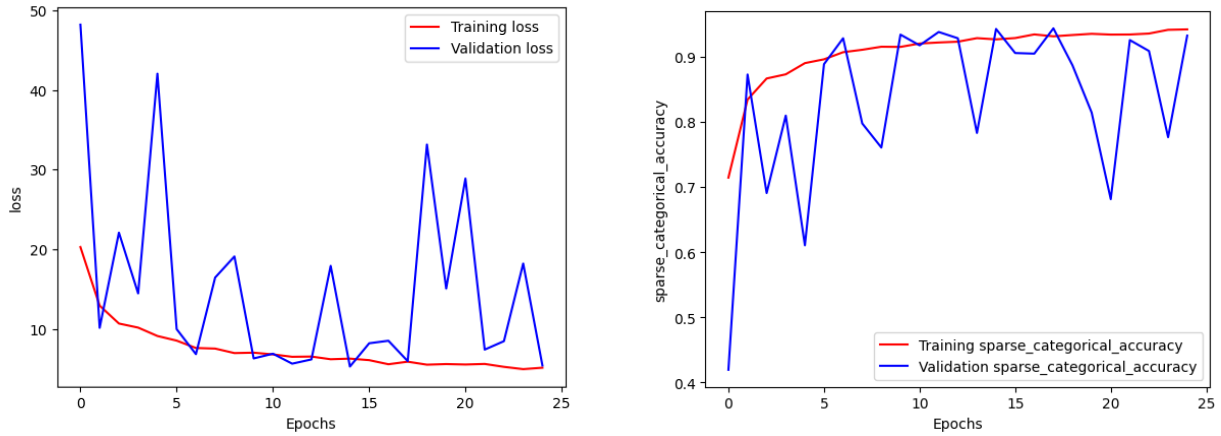
(a) Valores generados por la función de pérdida. (b) Valores del error absoluto medio generados.

Figura 4.1: Gráficas de métricas de función de pérdida y MAE generadas durante el entrenamiento del Autoencoder.

Ambas gráficas tienden a converger a los mismos valores. Mientras que los valores del conjunto de entrenamiento tienden a converger de manera abrupta durante los primeros *epochs* del entrenamiento, gracias a la optimización *Adam* la tasa de aprendizaje se ajusta y el valor obtenido por la función de pérdida disminuye de forma menos abrupta. Gracias a que utilizamos *early-stopping* el número de *epochs* es pequeño y se puede apreciar que los valores de validación no superan a los valores de entrenamiento, por lo que el modelo deja de entrenar en el momento óptimo, esto significa que nuestro modelo aprende con mucha rapidez y se llega a un punto mínimo dentro del descenso de gradiente.

Tras esto se realiza un proceso de *fine tuning*, en el cual se intercambian las capas que conforman el descodificador por una capa *soft-max*. Los pesos generados en las capas del codificador se mantienen, la tasa de aprendizaje se ha reducido a la hora de realizar el entrenamiento para evitar un sobreajuste. *Adam* se mantiene como algoritmo

de optimización mientras que la función de pérdida utilizada es la entropía cruzada categórica.



(a) Valores generados por la función de pérdida. (b) Valores de la métrica *sparse cat. accuracy*.

Figura 4.2: Gráficas de métricas de función de pérdida y *sparse categorical accuracy* generadas durante el entrenamiento del Autoencoder en la etapa de *fine tuning*.

Al contrario que en la etapa de entrenamiento de las gráficas de la Figura 4.1, los valores obtenidos a partir del conjunto de datos de validación generan valores oscilantes a lo largo de los *epochs* como se puede ver en las gráficas de la Figura 4.2, aunque durante el entrenamiento los valores generados por la función de pérdida convergen y los valores de precisión aumentan. Esto ocurre porque el modelo tiende a tener un sobreajuste si se utilizan datos tabulares como conjunto de entrenamiento y validación, ya que estos contienen datos de muy diferente naturaleza.

Accuracy	Precision	Recall	F1 Score
88.50 %	89,00 %	88,50 %	88,30 %

Tabla 4.3: Métricas de evaluación obtenidas a partir del conjunto de datos D4.

Aún así el modelo muestra métricas notables a la hora de predecir el diagnóstico con el conjunto de evaluación D4 *TADPOLE* como se puede observar en la Tabla 4.3, donde todos los valores son aproximadamente un 89 %.

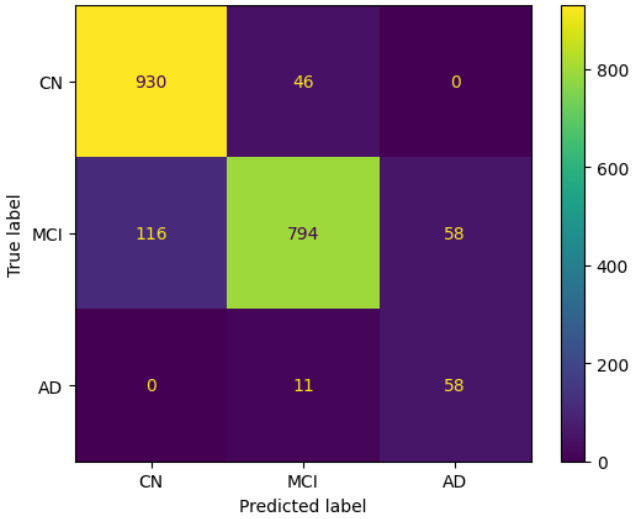


Figura 4.3: Matriz de confusión generada por el *autoencoder*.

La Figura 4.3 muestra la matriz de confusión, donde podemos observar que el *autoencoder* tiende a tener una predicción errónea con las parejas de clases CN-MCI y AD-MCI, en ambos casos tienden a predecir deterioro cognitivo leve en vez de cognitivamente normal o Alzheimer. Con respecto al diagnóstico MCI el *autoencoder* tiene una predicción que tiende a fallar más para el diagnóstico cognitivo normal que para la enfermedad de Alzheimer.

4.3. Comparación Redes Neuronales vs Autoencoder

La diferencia entre un *autoencoder* y un modelo basado en redes neuronales radica en que el *autoencoder* realiza una etapa de entrenamiento extra, ya que el primer entrenamiento se basa en un problema de reconstrucción del conjunto de datos de entrada y el segundo entrenamiento se basa en un problema de predicción de la clase a la que pertenece cada elemento del conjunto de datos. Se ha considerado realizar una comparativa para observar y analizar la predicción que tienen ambos modelos, también se han tenido en cuenta la ejecución del modelo *random forest*, el cual se ha utilizado también en etapas anteriores del proyecto.

Tanto el *autoencoder* como el modelo de redes neuronales utilizan el mismo conjunto de hiperparámetros, funciones de pérdida y algoritmos de optimización usados en etapas anteriores para que la comparación de los resultados sea lo más justa posible.

Tras entrenar a los modelos con el mismo conjunto de datos D1-D2 *TADPOLE* se han obtenido las siguientes métricas:

	Accuracy	Precision
Autoencoder	84.70 % ± 4.90 %	85.20 % ± 4.30 %
Redes Neuronales	82.10 % ± 4.90 %	82.20 % ± 5.80 %
Random Forest	**91.90 % ± 2.10 %**	**92.70 % ± 1.60 %**
	Recall	F1 Score
Autoencoder	86.70 % ± 3.80 %	85.00 % ± 4.70 %
Redes Neuronales	85.10 % ± 2,80 %	81.80 % ± 2.80 %
Random Forest	**91.70 % ± 2.40 %**	**92.10 % ± 2.10 %**

Tabla 4.4: Métricas obtenidas con los diferentes modelos a partir del conjunto D1-D2.

Las valores obtenidos para ambos modelos en la tabla de la figura 4.4 tienden a ser valores cercanos, aun así el *autoencoder* genera mejores valores que un modelo basado en redes neuronales ya que se realiza un entrenamiento extra. Sin embargo, el *autoencoder* tiene peores valores en comparación con los valores obtenidos por el *random forest*, por lo que los modelos de aprendizaje tradicionales tienden a generar mejores resultados que los modelos basados en redes neuronales cuando la entrada consiste en datos tabulares. Este resultado es conocido en la literatura reciente, donde modelos *Extreme Gradient Boosting* tienden a generar mejores resultados al evaluar los modelos *deep learning* con ciertos datos tabulares nuevos [17] [26].

4.4. Resultados de los modelos de Aprendizaje Tradicional con Autoencoder

Se ha planteado un estudio comparativo consistente en analizar las prestaciones de los modelos utilizando por un lado el modelo de aprendizaje tradicional, y por otro, el mismo modelo de aprendizaje usando un conjunto de datos cuya dimensionalidad se ha reducido gracias al *autoencoder*. También se han realizado diagramas de barras de las métricas de los modelos (*accuracy*, *precision*, *recall* y *f1 score*), curvas *Receiver Operating Characteristic (ROC)* y matrices de confusión. Tanto en las curvas ROC como en los diagramas de barras se han realizado comparaciones entre el modelo *baseline*, el autoencoder y el modelo con el conjunto de datos de dimensionalidad reducida.

A la hora de evaluar la predicción de cada uno de los modelos se ha utilizado el conjunto de D4 *TADPOLE* como conjunto de evaluación.

4.4.1. Comparación de las métricas del modelo

En primer lugar se compararon las métricas *accuracy*, *precision*, *recall* y *f1 score*, que se calcularon con la librería *Sklearn Metrics*. Debido a que el problema es un problema multiclase que se basa en tres clases diferentes se aplicó el método *weighted average*, que consiste en dividir en problema en varios problemas de clase binaria, se calcula la métrica correspondiente a esa clase y por último se suman las métricas obtenidas en los diferentes problemas estableciendo un peso a cada una, dicho peso depende de la representación que tiene esa misma clase dentro del conjunto de datos.

Los diagramas de barras de los modelos kNN, SVM, *random forest*, árboles de decisión y *gradient boosting* generados son los siguientes:

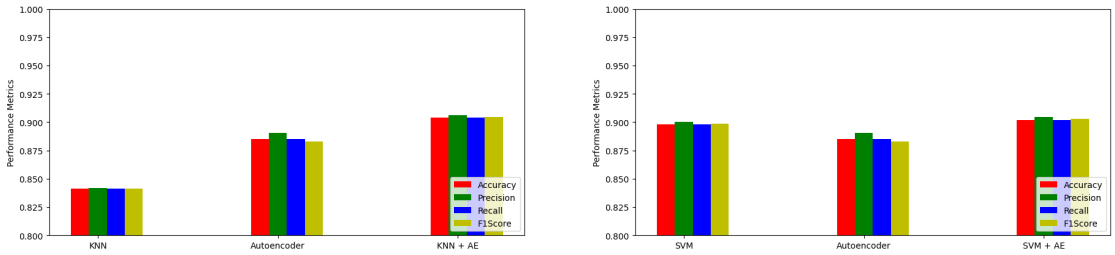


Figura 4.4: Diagrama de barras entre modelo kNN, SVM y *autoencoder*.

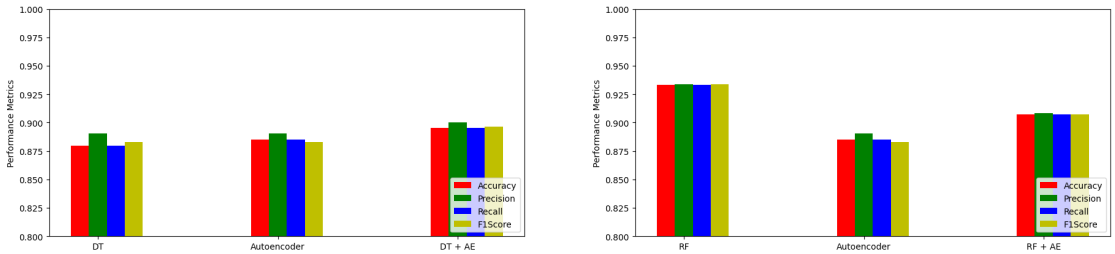


Figura 4.5: Diagrama de barras entre modelo árboles de decisión, *random forest* y *autoencoder*.

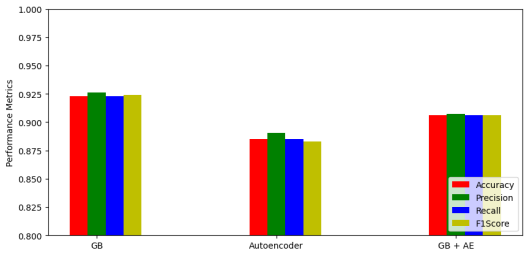


Figura 4.6: Diagrama de barras entre modelo *gradient boosting* y *autoencoder*.

Como se puede observar en los diagramas de barras de las Figuras 4.4-4.6, los modelos kNN y árboles de decisión tienden a generar valores por debajo del *autoencoder*,

mientras que el modelo SVM genera resultados ligeramente mejores. Al utilizar estos modelos en combinación con el *autoencoder* se produce una leve mejora en la predicción respecto a su modelo *baseline*, siendo el modelo que mayormente se beneficia de esta combinación es el modelo kNN. Respecto a los modelos *random forest* y *gradient boosting*, tienden a generar unos resultados mejores que el *autoencoder* y a la hora de combinarse se observan peores resultados. Por lo que se deduce de estos diagramas, si un modelo de aprendizaje tradicional tiene dificultades a la hora de generar buenas métricas de evaluación se puede utilizar un *autoencoder* como sistema de apoyo para obtener mejores resultados en la predicción.

4.4.2. Comparación de las curvas ROC

Para la comparación de los modelos se ha utilizado el método *micro averaging*, donde se agrega la contribución de cada una de las clases para calcular la media de cada métrica, calculando así el *True Positive Rate (TPR)* (4.1) como el *False Positive Rate (FPR)* (4.2).

$$TPR = \frac{\sum_c TP_c}{\sum_c (TP_c + FN_c)}, \tag{4.1}$$

$$FPR = \frac{\sum_c FP_c}{\sum_c (FP_c + TN_c)}, \tag{4.2}$$

En la tasa de verdaderos positivos se tienen en cuenta los verdaderos positivos (TP) y falsos negativos (FN), mientras que en la tasa de falsos positivos se tienen en cuenta los falsos positivos (FP) y verdaderos negativos (TN). Otra métrica a tener en cuenta es el *Area under the ROC Curve (AUC)*, que comprende valores entre 0.5 y 1, siendo 1 el valor diagnóstico completo y 0.5 un valor totalmente aleatorio.

Las curvas ROC generadas han sido las siguientes:

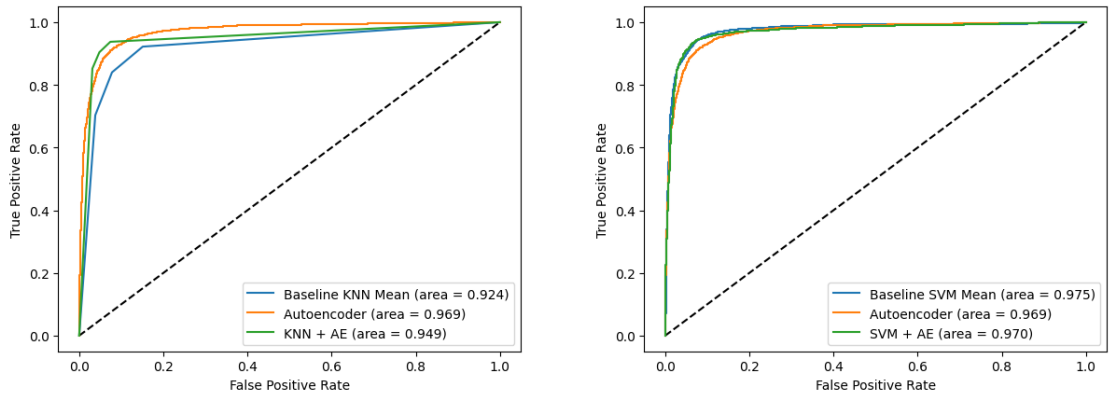


Figura 4.7: Curvas ROC entre modelo kNN, SVM y *autoencoder*.

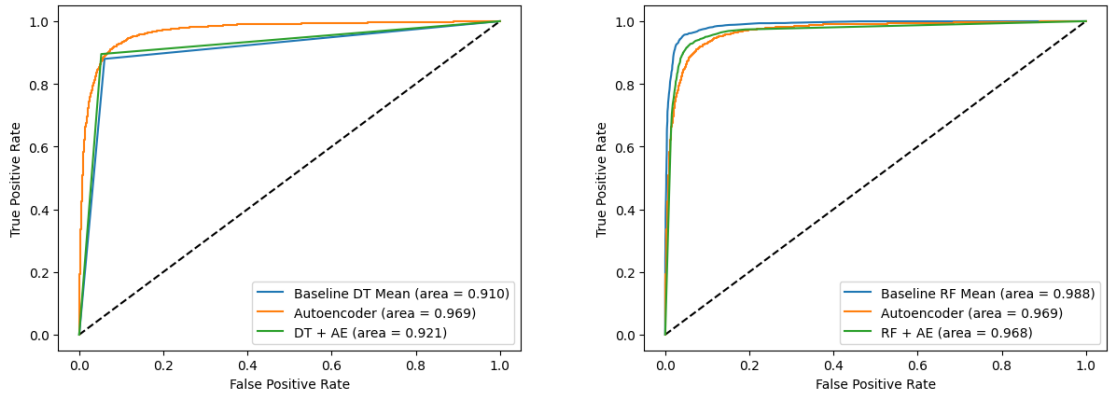


Figura 4.8: Curvas ROC entre modelo árboles de decisión, *random forest* y *autoencoder*.

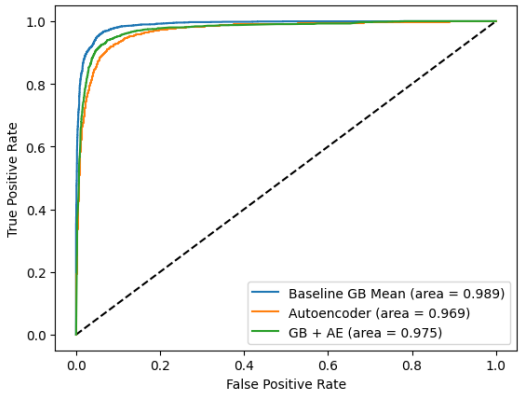


Figura 4.9: Curvas ROC entre modelo *gradient boosting* y *autoencoder*.

Todos los modelos muestran en las curvas ROC generadas en las Figuras 4.7-4.9 una buena fiabilidad al tener un valor AUC mayor a 0.9, en algunos modelos como kNN o árboles de decisión, el *autoencoder* demuestra tener un mayor valor AUC mientras que en otros casos los valores son iguales o muy parecidos. Se puede ver que en algunos casos el modelo combinado con el *autoencoder* mejora el área bajo la curva, en los modelos *random forest*, SVM y *gradient boosting* no es necesaria la intervención del *autoencoder*.

4.4.3. Coeficiente kappa de Cohen

Una de las métricas que también se ha tenido en cuenta en la evaluación es el coeficiente kappa de *Cohen*, es una métrica que se suele utilizar cuando las clases que conforman el conjunto de datos no están balanceadas. Esta métrica mide el grado de concordancia entre dos conjuntos de datos que están formados por valores categóricos. El coeficiente kappa de *Cohen* se calcula según la Ecuación 4.3.

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)}, \tag{4.3}$$

Donde $Pr(a)$ son los valores en los cuales están de acuerdos ambos observadores respecto a la totalidad de los valores estudiados y $Pr(e)$ es la probabilidad hipotética de acuerdo por azar, se calcula a partir de la suma de las filas y columnas de la matriz de confusión. El coeficiente kappa de *Cohen* va desde -1 hasta 1, siendo todo valor menor que 0 una fiabilidad nula o deficiente y 1 siendo una fiabilidad perfecta. En este problema utilizaremos como uno de los observadores el conjunto de datos de diagnósticos de los pacientes, mientras que el otro observador será el diagnóstico predicho por los modelos de aprendizaje.

	Modelo	Modelo con autoencoder
Autoencoder	0.78	
kNN	0.70	**0.82**
SVM	0.80	**0.81**
DT	0.77	**0.80**
RF	**0.87**	0.82
GB	**0.85**	0.82

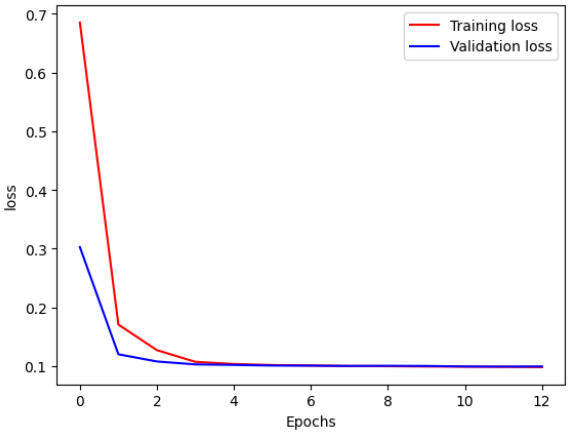
Tabla 4.5: Coeficiente kappa de *Cohen* obtenidos por los modelos

Los modelos que mejor coeficiente de kappa de *Cohen* tienen son *random forest* y *gradient boosting*, cuyos valores mayores o equivalentes a 0.85. El autoencoder genera un coeficiente similar al modelo *decision tree*, siendo el cuarto mejor modelo. Cuando estos modelos se combinan con el autoencoder se observa una mejora con respecto a su modelo *baseline* excepto en los casos de *gradient boosting* y *random forest*, aún así, según la escala de *Landis* y *Koch* [27] los modelos varían entre casi perfectos y sustanciales al ser valores mayores a 0.8.

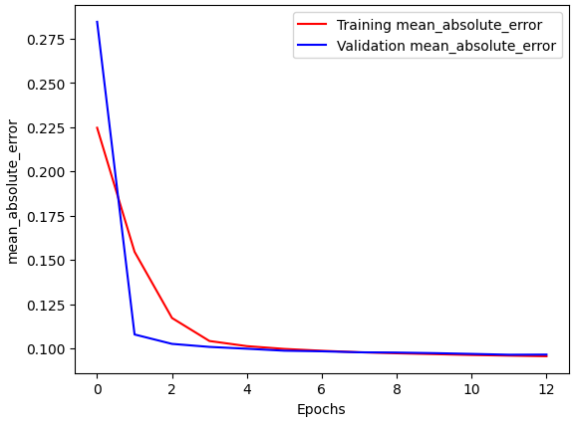
4.5. Predicción en el problema sMCI/pMCI

Para el problema de clasificación sMCI/pMCI, la función de pérdida es la misma que se ha utilizado en el problema de clasificación CN/MCI/AD, aunque se podría realizar un cambio a la función de pérdida *Binary Cross Entropy (BCE)* ya que el diagnostico se conforma por clases binarias, pero gracias a que la función entropía cruzada categórica es la misma que la función entropía cruzada binaria pero estableciendo un número de clases equivalente a dos, no será necesario realizar el cambio. Se han utilizado los mismos hiper-parámetros y también se ha realizado un proceso de *fine tuning* para predecir el diagnóstico de cada paciente al igual que para el problema anterior.

Los resultados del entrenamiento con el conjunto de datos D1-D2 son los siguientes:



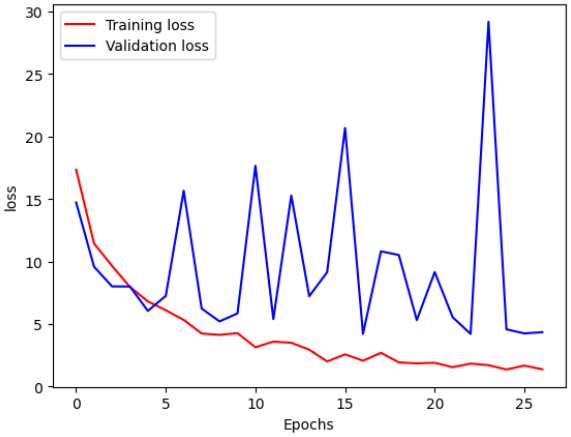
(a) Valores generados por la función de pérdida.



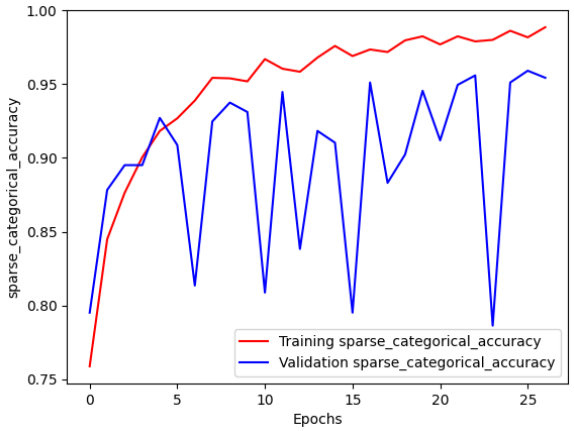
(b) Valores de la métrica *mean absolute error*.

Figura 4.10: Gráficas de función de pérdida y MAE generadas durante el entrenamiento del *autoencoder*.

El comportamiento del descenso de gradiente generado en estas gráficas de la Figura 4.10 es parecido a las gráficas de la Figura del problema anterior 4.1, en este caso la cantidad de *epochs* ejecutados es mayor.



(a) Valores generados por la función de pérdida.



(b) Valores de la métrica *sparse cat. accuracy*.

Figura 4.11: Gráficas de función de pérdida y *sparse categorical accuracy* generadas durante el entrenamiento del *autoencoder* en la etapa *fine tuning*.

Las gráficas de este entrenamiento de la Figura 4.11 y las gráficas del problema CN/MCI/AD de la Figura 4.2 tienen un comportamiento parecido, sólo que en este caso el descenso es más abrupto.

Accuracy	Precision	Recall	F1 Score
94.30 %	95,00 %	94,30 %	94,60 %

Tabla 4.6: Métricas de evaluación obtenidas a partir del conjunto de datos D4.

En este problema, las métricas de evaluación con el conjunto de datos D4 en la tabla de la Figura 4.6 son mejores que en el problema anterior, obteniendo valores aproximados a 95 %.

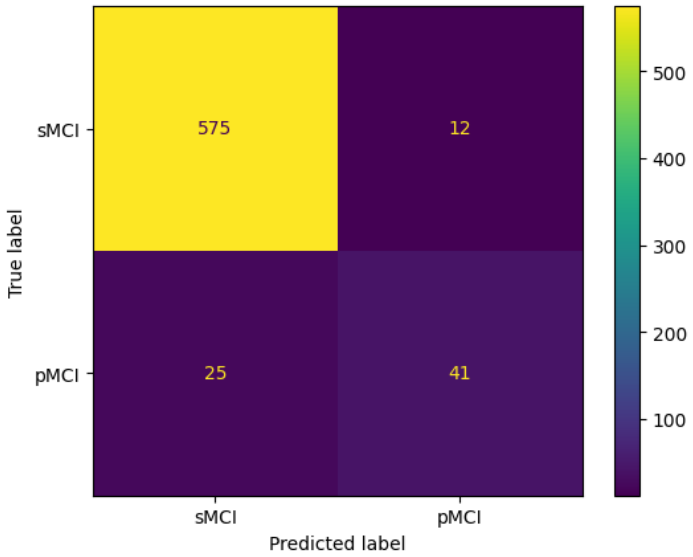


Figura 4.12: Matriz de confusión generada por el *autoencoder*.

En la matriz de confusión de la Figura 4.12 podemos observar que el *autoencoder* tiende a predecir erróneamente la clase pMCI, mientras que respecto a la clase sMCI tiene un número reducido de predicciones erróneas. En caso de existir más valores correspondientes a la clase pMCI dentro del conjunto de datos de entrenamiento, el *autoencoder* tendría una mayor certeza a la hora de predecir esta clase.

4.5.1. Resultados en la predicción de diagnóstico sMCI/pMCI

Al igual que en el problema anterior, se realizaron comparaciones entre diferentes implementaciones de los modelos mediante la generación de gráficos de barras de las métricas, curvas ROC y coeficiente kappa de *Cohen*.

Se ha usado el conjunto D4 TADPOLE para generar los resultados obtenidos por cada modelo. A la hora de calcular las métricas (*precision*, *recall* y *f1 score*) también se ha utilizado el método *weighted average*, ya que el conjunto de evaluación es un conjunto desbalanceado.

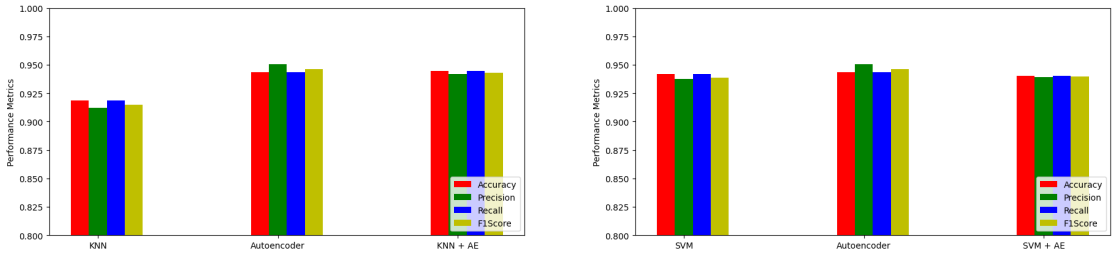


Figura 4.13: Diagrama de barras entre modelo kNN, SVM y *autoencoder*.

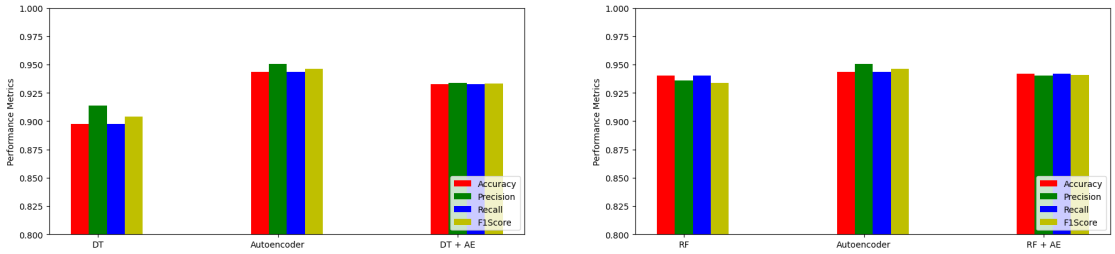


Figura 4.14: Diagrama de barras entre modelo árboles de decisión, *random forest* y *autoencoder*.

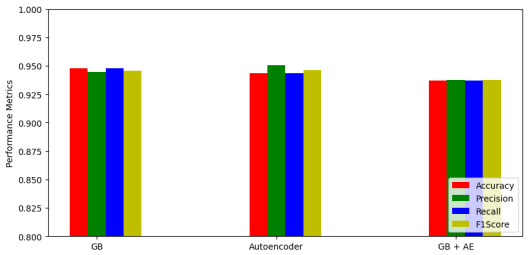


Figura 4.15: Diagrama de barras entre modelo *gradient boosting* y *autoencoder*.

En las gráficas de la predicción de diagnóstico sMCI/pMCI de las Figuras 4.13-4.15, el *autoencoder* genera mejores métricas que cualquiera de los modelos restantes, en el caso del modelo *gradient boosting* se observan unos valores similares. A la hora de combinar el *autoencoder* con los demás modelos se observa una mejora en los valores del modelo *baseline*, aproximándose estos al *autoencoder*. En este caso el modelo *decision tree* no predice con la misma certeza que los demás modelos siendo el que peor predice el conjunto de datos.

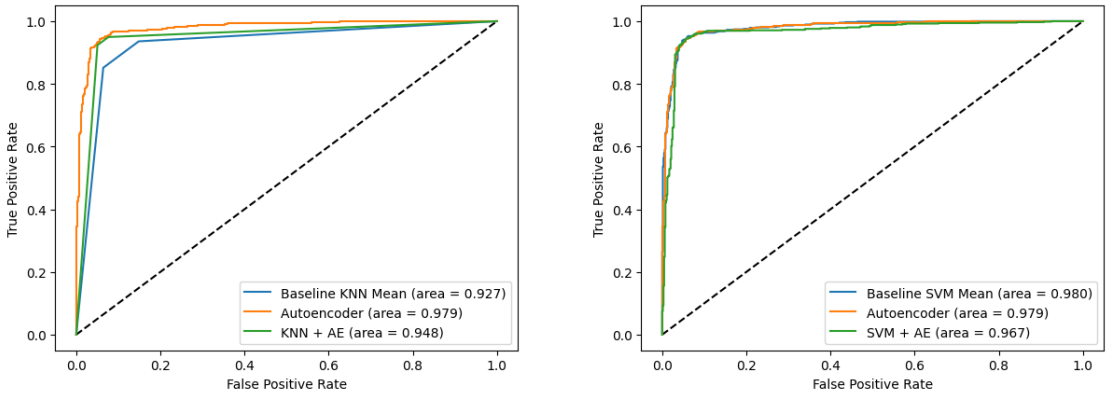


Figura 4.16: Curvas ROC entre modelo kNN, SVM y *autoencoder*.

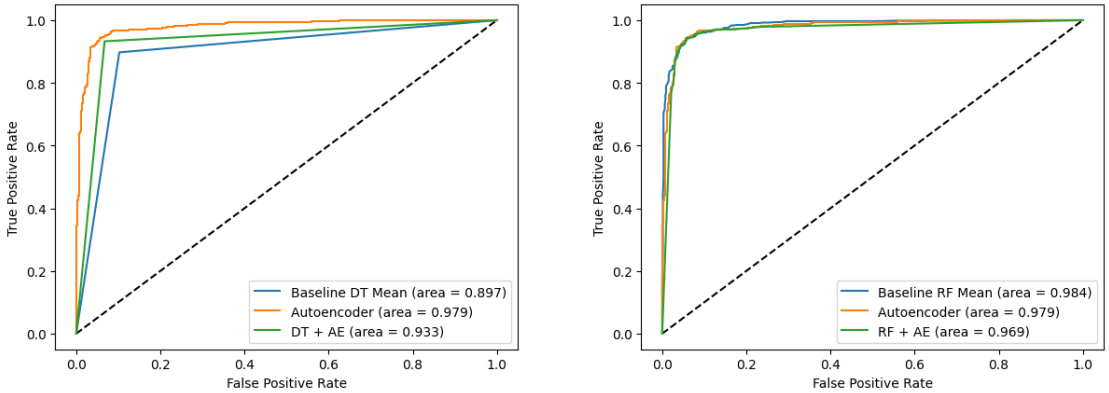


Figura 4.17: Curvas ROC entre árboles de decisión, *random forest* y *autoencoder*.

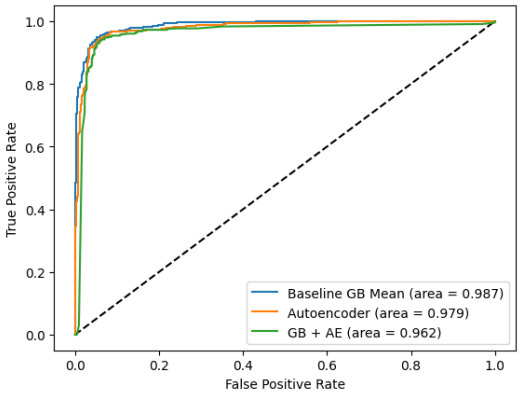


Figura 4.18: Curvas ROC entre modelo *gradient boosting* y *autoencoder*.

En las curvas ROC generadas en las Figuras 4.16-4.18 los modelos que peor valor AUC generan son el modelo kNN y el modelo *decision tree*, al igual que con el diagnóstico anterior cuando se combina el modelo con el *autoencoder* generan un mejor valor AUC. El comportamiento de la curvas ROC en los modelos SVM, *gradient boosting* y *random forest* es mejor comparado con el *autoencoder*, y el valor AUC generado por el modelo

basado en *autoencoder* no supera al valor generado por el *autoencoder* en ninguno de los cinco modelos.

	Modelo	Modelo con autoencoder
Autoencoder	0.65	
kNN	0.51	**0.67**
SVM	0.64	**0.66**
DT	0.51	**0.63**
RF	0.60	**0.67**
GB	**0.68**	0.65

Tabla 4.7: Coeficiente kappa de *Cohen* en la predicción sMCI/pMCI.

Por último, los coeficientes obtenidos en la Tabla 4.7 son peores a los coeficientes de la Tabla 4.5 de la anterior predicción, donde todos los valores eran mayores a 0.70, en este caso los valores varían entre 0.50 y 0.70. Los modelos *SVM* y *gradient boosting* son los que mejor coeficiente generan. También se obtiene una mejora en aquellos modelos con menor coeficiente a la hora de utilizar un *autoencoder* como modelo de apoyo, como es en el caso del modelo kNN y el modelo *decision tree*. Los modelos tienen una estimación moderada o buena según la escala de *Landis* y *Koch*.

4.6. Análisis de las métricas de evaluación entre modelo baseline y modelo basado en autoencoder

También se ha realizado un análisis de las métricas de evaluación entre los distintos tipos de modelos de aprendizaje tradicional, al igual que en las pruebas anteriores, se ha usado tanto el conjunto de datos original como el conjunto de datos reducido previamente por el *autoencoder*.

Como se puede ver en las Tablas 4.8-4.9, al combinar el *autoencoder* con los modelos tradicionales para la predicción CN/MCI/AD podemos observar que el modelo basado en *autoencoder* tiende a disminuir en pequeña medida el valor de las métricas, esto se puede apreciar en los modelos *random forest* y *gradient booster*, ya que ambos modelos son los que mejor resultado generan sin necesidad de reducir la dimensionalidad de la entrada. Los restantes modelos tienen una mejoría en las métricas, donde sus valores aumentan hasta un 90 %, el modelo que más se beneficia de aplicar el *autoencoder* el modelo kNN donde se pasa de un 84 % a un 90 %. Con respecto a la predicción del diagnóstico sMCI/pMCI, el único modelo que sale perjudicado es el *gradient booster*, mientras que los demás modelos mejoran al aplicar el *autoencoder*.

	Sin autoencoder				Con autoencoder			
	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
kNN	0.841	0.842	0.841	0.841	**0.900**	**0.901**	**0.900**	**0.900**
SVM	0.898	0.900	0.898	0.899	**0.899**	**0.901**	**0.899**	**0.900**
DT	0.877	0.888	0.877	0.88	**0.889**	**0.893**	**0.889**	**0.891**
RF	**0.935**	**0.935**	**0.935**	**0.935**	0.899	0.900	0.899	0.899
GB	**0.923**	**0.926**	**0.923**	**0.924**	0.898	0.900	0.898	0.899

Tabla 4.8: Métricas de evaluación en predicción CN/MCI/AD entre distintos modelos.

	Sin autoencoder				Con autoencoder			
	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
kNN	0.919	0.912	0.919	0.915	**0.945**	**0.942**	**0.945**	**0.943**
SVM	**0.942**	0.938	**0.942**	0.939	0.94	**0.939**	0.94	**0.94**
DT	0.897	0.914	0.897	0.904	**0.933**	**0.934**	**0.933**	**0.933**
RF	0.94	0.936	0.94	0.934	**0.942**	**0.94**	**0.942**	**0.941**
GB	**0.948**	**0.945**	**0.948**	**0.946**	0.937	0.938	0.937	0.937

Tabla 4.9: Métricas de evaluación en predicción sMCI/pMCI entre distintos modelos.

Los modelos que mejor resultado tienen en el primer problema al no usarse el *autoencoder* son *random forest* y *gradient booster*, con respecto al segundo problema, únicamente *gradient booster* es consistente, siendo de los mejores modelos junto al método SVM, aunque se puede observar que tanto *random forest* como *gradient booster* tienen una reducción de las métricas si el *autoencoder* influye, y por lo tanto, métodos como kNN y SVM tienden a ser mejores y consistentes en ambos problemas en este caso, también es notable mencionar que el método SVM es el único con muy baja influencia del *autoencoder* con respecto a la mejora de sus métricas, aumentando levemente estas en ambos problemas. Al aplicarse el *autoencoder* a los modelos de aprendizaje tradicional se puede ver que las métricas de evaluación son muy parecidas entre ellos tanto con la predicción del diagnóstico CN/MCI/AD como con la predicción del diagnóstico sMCI/pMCI, obteniéndose valores entre 0.900 % y 0.942 %.

4.6.1. Análisis del tiempo de ejecución entre modelos de aprendizaje tradicional y modelos basados en autoencoder

Por último, se ha realizado un análisis del tiempo de ejecución que existe en el proceso de entrenamiento de cada uno de los modelos, tanto con el conjunto de datos D1-D2 de *TADPOLE* como con el conjunto reducido en ambos problemas de diagnóstico.

Las tablas de tiempo obtenidas son las siguientes:

	KNN	SVM	DT	RF	GB
Baseline	0.062	62.29	17.72	45.32	99.77
Baseline + autoencoder	0.002	5.27	1.44	8.30	27.94

Tabla 4.10: Comparación de tiempo de ejecución en segundos entre modelos en el diagnostico CN/MCI/AD.

	KNN	SVM	DT	RF	GB
Baseline	0.03	10.85	6.64	17.38	19.83
Baseline + autoencoder	0.0009	0.329	0.336	1.88	3.98

Tabla 4.11: Comparación de tiempo de ejecución en segundos entre modelos en el diagnostico sMCI/pMCI.

Como se puede apreciar en las Tablas 4.10 y 4.11, el tiempo disminuye notablemente en cada uno de los modelos si se le aplica como entrada un conjunto de datos reducido por el *autoencoder*, ya que se está utilizando un conjunto de datos que contiene un total de 150 atributos en vez de un conjunto que tiene unos 750 atributos. Tanto estos resultados como los ya mostrados anteriormente nos indican que utilizar un modelo que se base en reducir la dimensionalidad del problema es positivo, ya que se obtiene una mejora en la precisión respecto a no usarlo, además de un entrenamiento más rápido del conjunto de datos.

Capítulo 5

Conclusiones

El objetivo de este trabajo consiste en analizar el comportamiento de un modelo *autoencoder* con datos tabulares, utilizando como referencia el artículo *Multimodal deep learning models for early detection of Alzheimer's disease stage* realizado por *Janani Venugopalan et al* [15]. Los datos de entrenamiento y validación con los que se ha trabajado en este proyecto se han obtenido del *TADPOLE CHALLENGE*. Se ha diseñado un *autoencoder*, el cual realiza una codificación y decodificación los datos de entrada, al que después se le ha realizado un proceso de ajuste fino para poder predecir la clase correspondiente a cada uno de los datos de entrada, para más tarde obtener un modelo que se encarga de reducir la dimensionalidad y que sirve como apoyo a los modelos de aprendizaje tradicional.

Utilizando un modelo *random forest* se han conseguido unos valores de precisión y *F1 Score* similares a los resultados del mismo modelo usado en el artículo de *Shaker El Sappagh et al "A multilayer multimodal detection and prediction model based on explainable artificial intelligence for Alzheimer's disease*" [25] donde nosotros obtenemos aproximadamente 90 % de precisión y valor *F1 score* en alguna combinación de las modalidades, mientras que en otras obtenemos mejores valores que los que se ven en el artículo. Esto nos permite obtener métodos reproducibles y acordes con el estado del arte.

Además, se ha realizado una comparación entre un *autoencoder* y un modelo basado en redes neuronales. Tras entrenar ambos modelos utilizando los mismos hiper parámetros se ha demostrado que el modelo *autoencoder* es ligeramente mejor que el modelo basado en redes neuronales debido a los resultados obtenidos, pero ambos modelos no superan al modelo *random forest* del trabajo de *El Sappagh et al* [25].

Tras obtener un *autoencoder* totalmente funcional se ha procedido a estudiar el

comportamiento del modelo a la hora de utilizar como datos de entrada un conjunto de datos tabulares, se ha realizado una comparación entre un modelo *autoencoder*, un modelo de aprendizaje tradicional y la combinación de ambos, mostrando así unos resultados similares entre ellos. Gracias al *autoencoder* estamos introduciendo una menor dimensionalidad en los datos de entrada por lo cual el tiempo de ejecución de los modelos de aprendizaje tradicional es menor y también se obtiene una ligera mejora en la precisión de ciertos modelos de aprendizaje tradicional a la hora de realizar una predicción del diagnóstico en modelos como kNN, SVM y árboles de decisión.

Un problema que se ha comprobado en este proyecto es la necesidad de una gran cantidad de datos a la hora de entrenar un modelo *autoencoder*, como es en el caso del problema de diagnóstico sMCI/pMCI, y que también este tipo de modelos tiene dificultades a la hora de entrenar un conjunto de datos tabulares y no predice con tanta precisión como otro tipo de modelos de aprendizaje tradicional.

Como futuros proyectos, se podría completar el conjunto de datos usado en este proyecto, es decir, utilizar datos de tipo genético, clínico e imágenes MRI y analizar el resultado que se obtiene de estos modelos a la hora de entrenar con estos conjuntos de datos de manera individual como es en el caso del artículo de *Janani Venugopalan et al* [15] que se ha tomado como referencia, otra sugerencia sería la de extender la temática del proyecto a otro tipo de enfermedades neurodegenerativas como el Parkinson, comprobando así cual sería la capacidad del modelo basado en *autoencoder*, o utilizar otros tipos de modelos más enfocados a *deep learning* para poder estudiar el comportamiento que tienen a la hora de combinar estos modelos con otros.

Capítulo 6

Bibliografía

[1] 2023 Alzheimer's disease facts and figures. *Epub*, 2023.

[2] Anil Kumar et al. Alzheimer disease. *StatPearls*, 2022.

[3] National Collaborating Centre for Mental Health. *Dementia: Quick reference guide*. National Institute for Health and Clinical Excellence., 2008.

[4] Fernando Durães et al. Old drugs as new treatments for neurodegenerative diseases vol. 11,2 44. *Frontiers in Neurorobotics*, 2018.

[5] Moradi Seyed et al. Nanoformulations of herbal extracts in treatment of neurodegenerative disorders. *Frontiers in Bioengineering and Biotechnology*, 2020.

[6] P. M. Rodrigues et al. Lacsogram: A new eeg tool to diagnose Alzheimer's disease. *IEEE Journal of Biomedical and Health Informatics, vol. 25, no. 9, pp. 3384-3395*, 2021.

[7] Chong-Yaw Wee. Prediction of Alzheimer's disease and mild cognitive impairment using cortical morphological patterns. *Hum. Brain Mapp*, 2013.

[8] A P Porsteinsson et al. Diagnosis of early Alzheimer's disease: Clinical practice in 2021. *The journal of prevention of Alzheimer's disease vol. 8,3*, 2021.

[9] Jusi Mattila. A disease state fingerprint for evaluation of Alzheimer's disease. *J. Alzheimer's Dis*, 2011.

[10] Simeon Spasov et al. A parameter-efficient deep learning approach to predict conversion from mild cognitive impairment to Alzheimer's disease. *NeuroImage vol. 189*, 2019.

[11] Yiming Ding, Jae Ho Sohn, Michael G. Kawczynski, Hari Trivedi, Roy Harnish, Nathaniel W. Jenkins, Dmytro Lituiev, Timothy P. Copeland, Mariam S. Aboian,

Carina Mari Aparici, Spencer C. Behr, Robert R. Flavell, Shih-Ying Huang, Kelly A. Zalocusky, Lorenzo Nardo, Youngho Seo, Randall A. Hawkins, Miguel Hernandez Pampaloni, Dexter Hadley, and Benjamin L. Franc. A deep learning model to predict a diagnosis of Alzheimer disease by using 18f-fdg pet of the brain. *Radiology*, 290(2):456–464, 2019. PMID: 30398430.

[12] Sumit Sharma. Heart diseases prediction using deep learning neural network model. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 2020.

[13] Subrato Bharati. Hybrid deep learning for detecting lung diseases from x-ray images. *Informatics in Medicine Unlocked*, 2020.

[14] Arifa Shikalgar. Hybrid deep learning approach for classifying Alzheimer disease based on multimodal data. *Computing in Engineering and Technology*, 2020.

[15] Janani Venugopalan et al. Multimodal deep learning models for early detection of Alzheimer's disease stage. *Sci Rep*, 2021.

[16] Tadpole Grand-Challenge. https://tadpole.grand-challenge.org/.

[17] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[18] Ping Li and Phan-Minh Nguyen. On random deep weight-tied autoencoders: Exact asymptotic analysis, phase transitions, and implications to training. In *International Conference on Learning Representations*, 2019.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[20] Zhongheng Zhang. Introduction to machine learning: k-nearest neighbors. *Ann Transl Med*, 2016.

[21] Derek A. Pisner et al. Machine learning - chapter 6 - support vector machine. *Academic Press*, 2020.

[22] Yan-yan Song. Decision tree methods: applications for classification and prediction. *Shanghai Arch Psychiatry*, 2015.

[23] Steven J. Rigatti. *Random Forest.* J Insur Med, 2017.

[24] Natekin Alexey. Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, 2013.

[25] Shaker El-Sappagh et al. A multilayer multimodal detection and prediction model based on explainable artificial intelligence for Alzheimer's disease. *Sci Rep*, 2021.

[26] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2022.

[27] JR Landis and GG Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159—174, March 1977.

Lista de Figuras

Lista de Tablas

Anexos

Anexos A

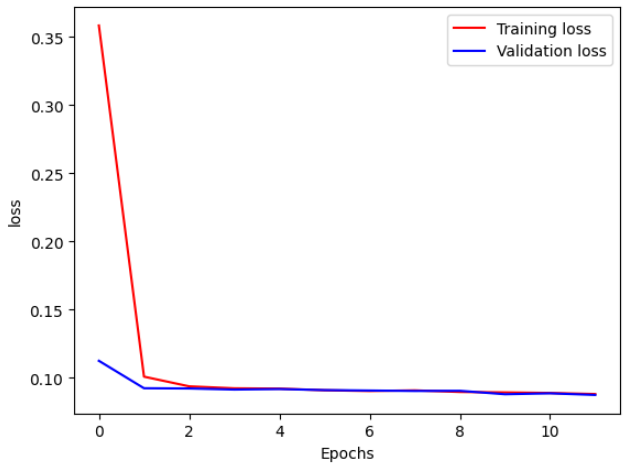
Recopilación de los resultados

A.1. Conjunto de datos: TADPOLE Challenge

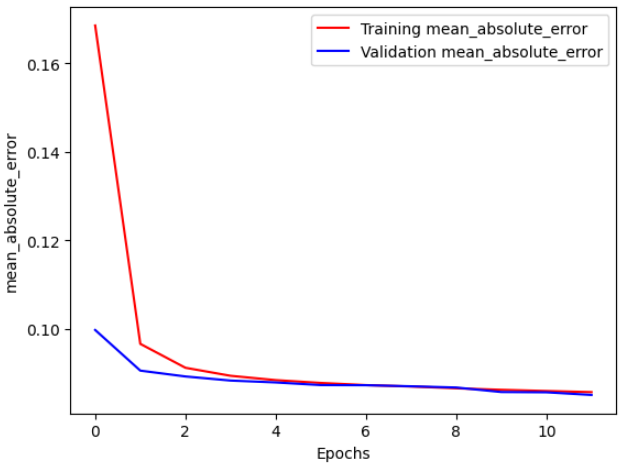
Se ha utilizado el conjunto de datos *TADPOLE Challenge* en este proyecto para comprobar reproducibilidad de los modelos generados. Tras realizar una etapa de preprocesado para generar los datos de entrenamiento y evaluación se han obtenido un total de 12.726 pruebas clínicas de diferentes pacientes, de las cuales 4087 son cognitivamente normales, 5579 tienen un diagnóstico de deterioro cognitivo leve y 3060 se les ha diagnosticando con Alzheimer. Por otro lado se ha creado otro conjunto de datos del mismo origen sólo teniendo en cuenta aquellos pacientes con deterioro cognitivo leve, dividiéndose así en deterioro cognitivo leve estable y deterioro cognitivo leve progresivo, siendo un total de 4815 pacientes de los cuales se dividen en 3420 con un diagnóstico sMCI y 1395 con un diagnóstico pMCI.

A.2. Resultados predicción AD/MCI/CN

A.2.1. Gráficas generadas en la etapa de entrenamiento y fine tuning del autoencoder

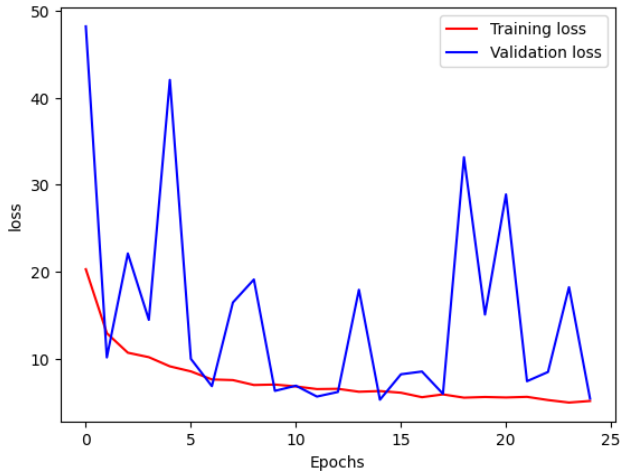


(a) Valores generados por la función de pérdida.

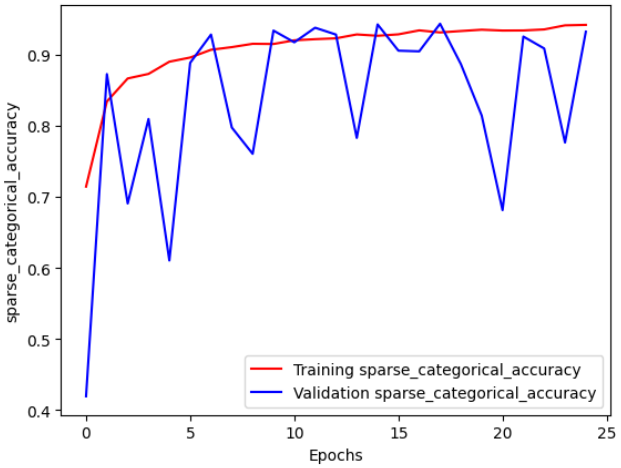


(b) Valores del error absoluto medio generados.

Figura A.1: Gráficas de métricas de función de pérdida y MAE generadas durante el entrenamiento del Autoencoder.



(a) Valores generados por la función de pérdida.



(b) Valores de la métrica *sparse categorical accuracy*.

Figura A.2: Gráficas de métricas de función de pérdida y *sparse categorical accuracy* generadas durante el entrenamiento del Autoencoder en la etapa *fine tuning*.

A.2.2. Matrices de confusión

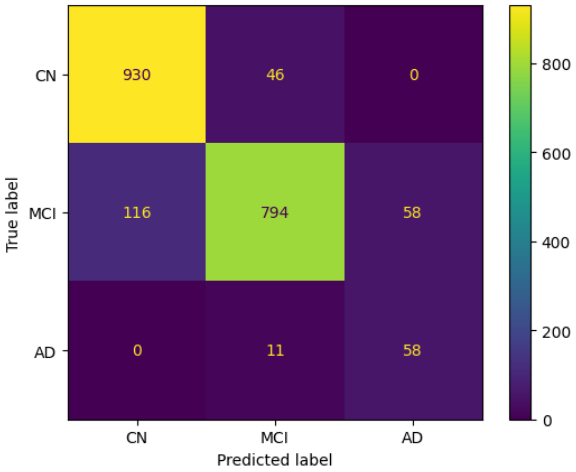
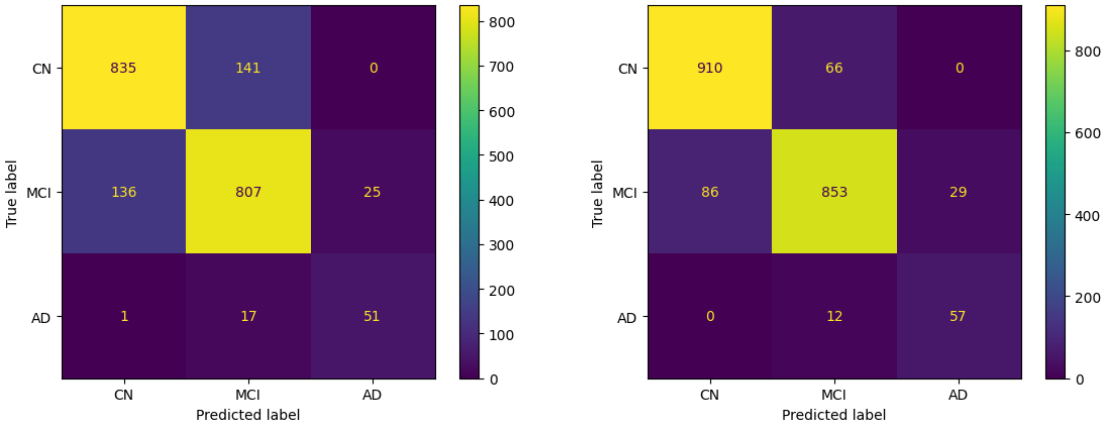
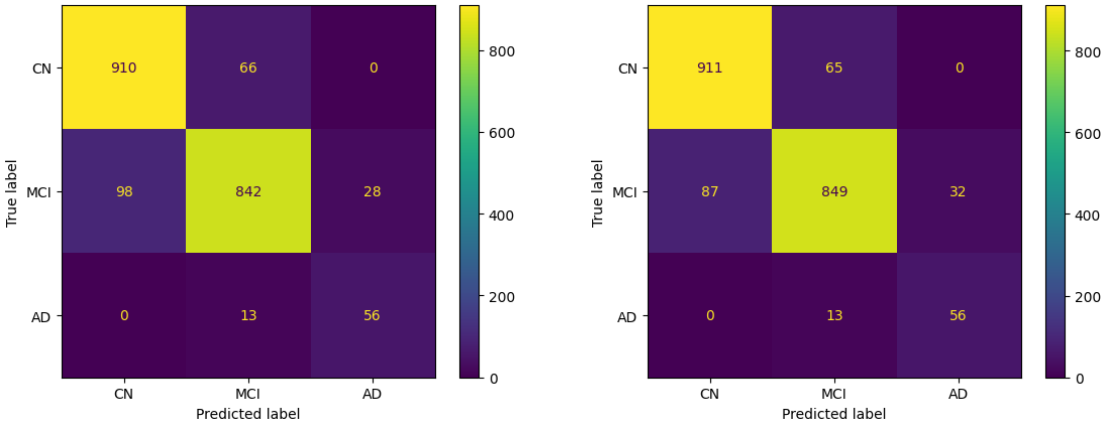


Figura A.3: Matriz de confusión del autoencoder.



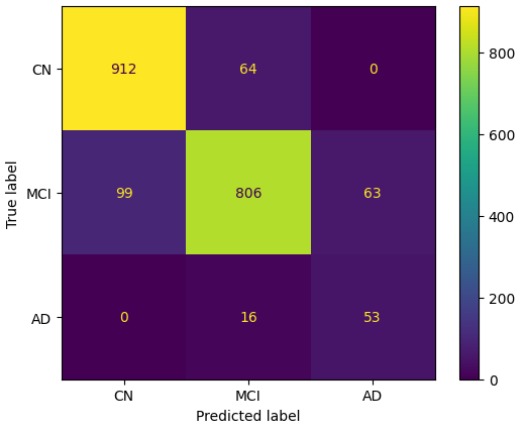
(a) Matriz de confusión modelo kNN. (b) Matriz de confusión modelo kNN + AE.

Figura A.4: Matrices de confusión entre kNN y *autoencoder*.

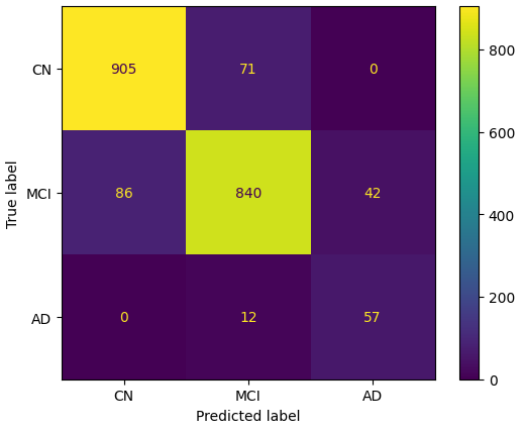


(a) Matriz de confusión modelo SVM. (b) Matriz de confusión modelo SVM + AE.

Figura A.5: Matrices de confusión entre SVM y *autoencoder*.

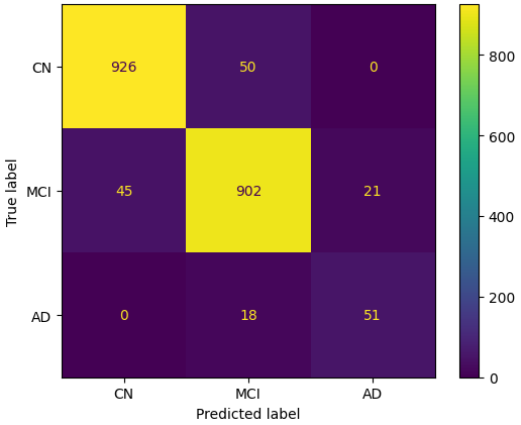


(a) Matriz de confusión modelo DT.

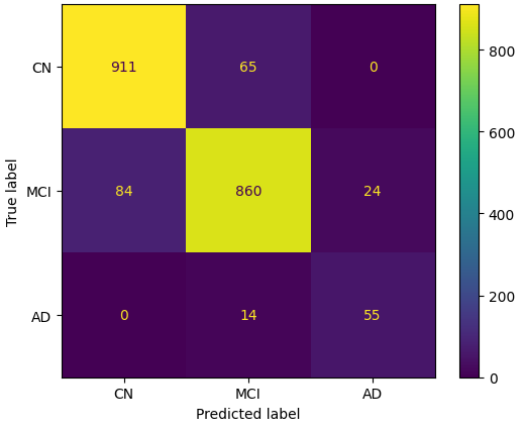


(b) Matriz de confusión modelo DT + AE.

Figura A.6: Matrices de confusión entre DT y *autoencoder*.

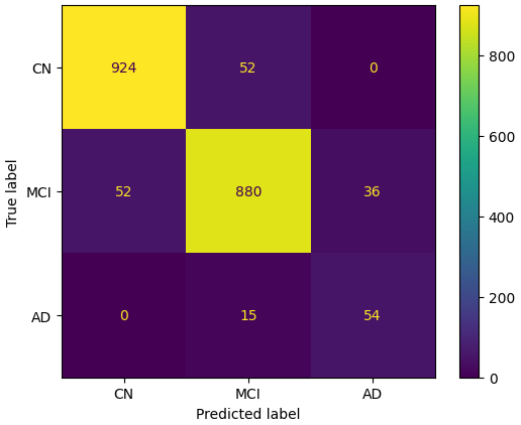


(a) Matriz de confusión modelo RF.

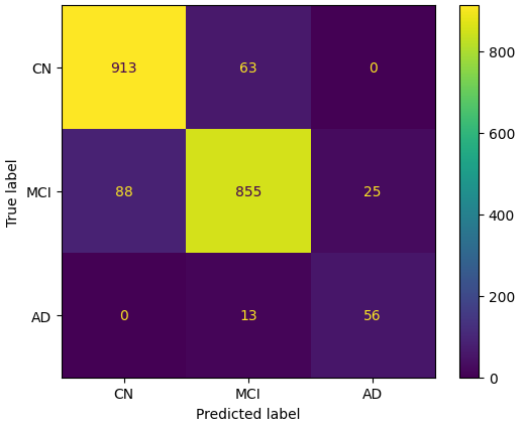


(b) Matriz de confusión modelo RF + AE.

Figura A.7: Matrices de confusión entre RF y *autoencoder*.



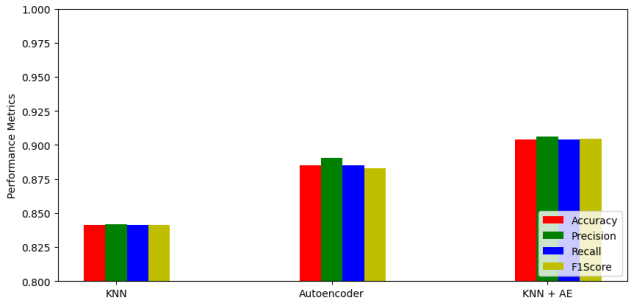
(a) Matriz de confusión modelo GB.



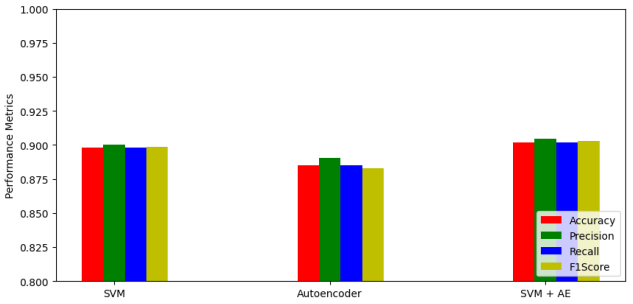
(b) Matriz de confusión modelo GB + AE.

Figura A.8: Matrices de confusión entre GB y *autoencoder*.

A.2.3. Métricas de evaluación entre modelos

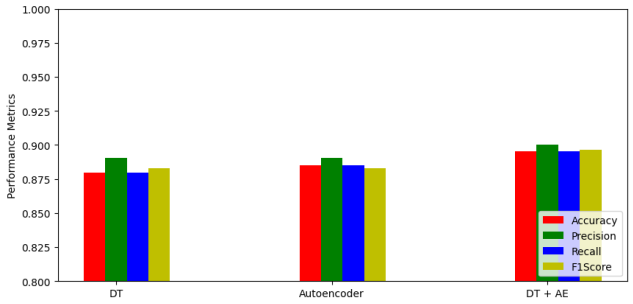


(a) Comparación de métricas de modelo kNN.

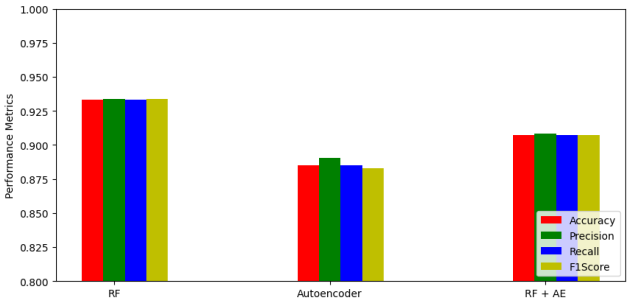


(b) Comparación de métricas de modelo SVM.

Figura A.9: Métricas de modelo kNN y SVM.



(a) Comparación de métricas de modelo DT.



(b) Comparación de métricas de modelo RF.

Figura A.10: Métricas de modelo DT y RF.

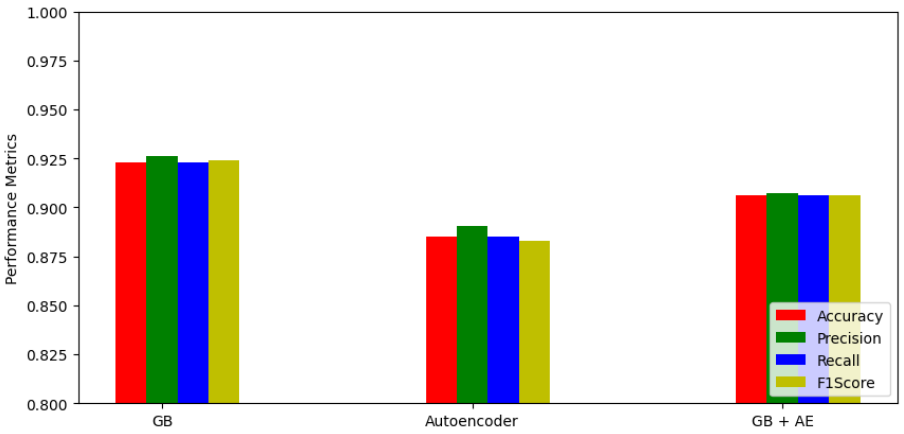


Figura A.11: Comparación de métricas de modelo GB.

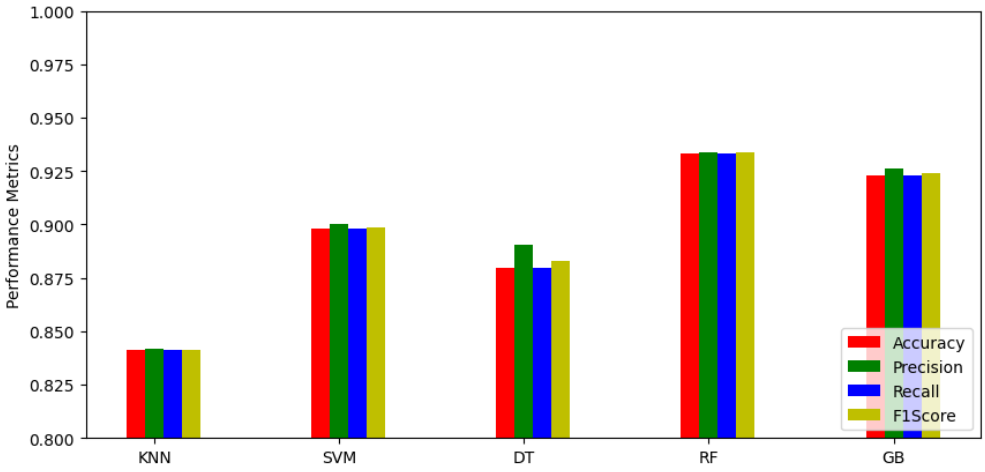


Figura A.12: Comparación de métricas entre modelos superficiales.

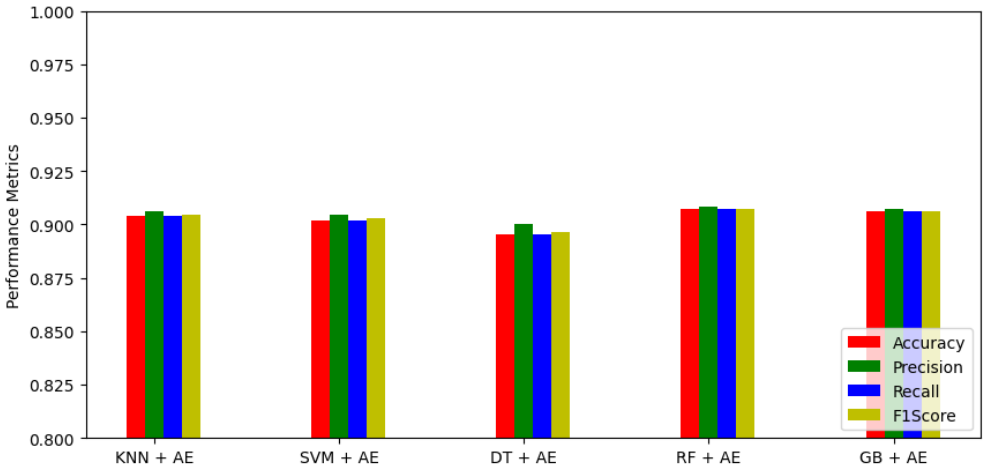
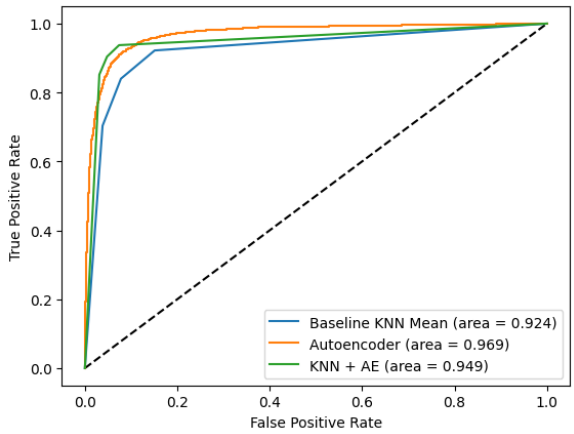
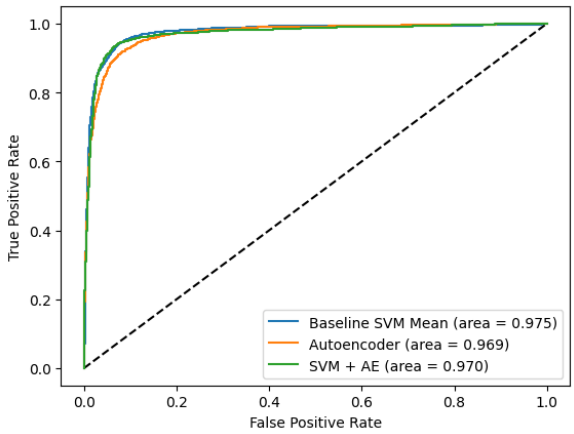


Figura A.13: Comparación de métricas entre modelos superficiales basados en *autoencoder*.

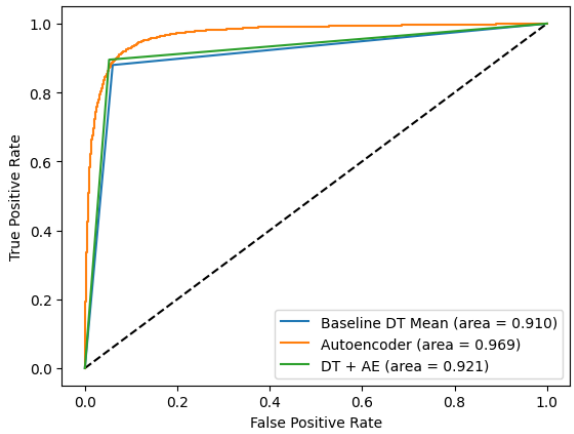
A.2.4. Curvas ROC entre modelos

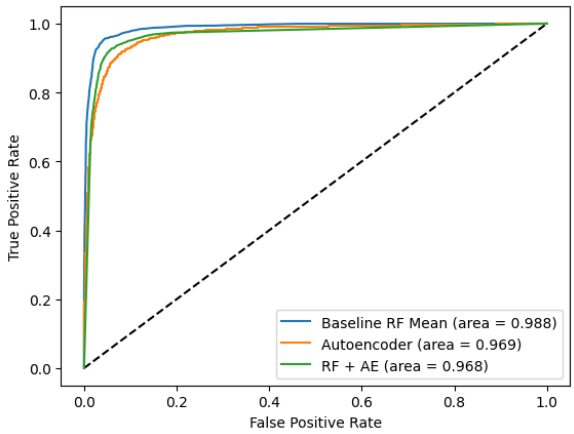




(a) Comparación de curvas ROC de modelo kNN. (b) Comparación de curvas ROC de modelo SVM.

Figura A.14: Métricas de modelo kNN y SVM.





(a) Comparación de curvas ROC de modelo DT. (b) Comparación de curvas ROC de modelo RF.

Figura A.15: Métricas de modelo DT y RF.

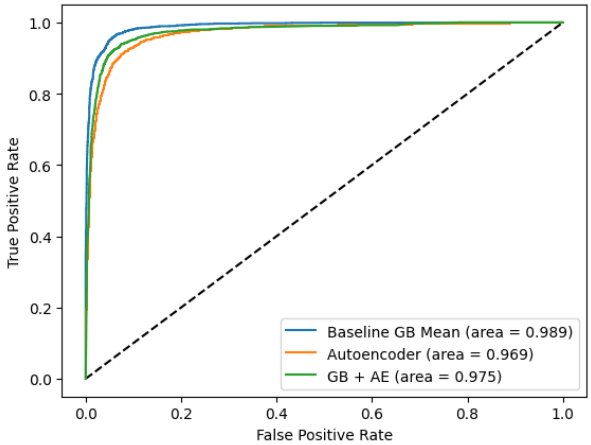


Figura A.16: Comparación de curvas ROC de modelo GB.

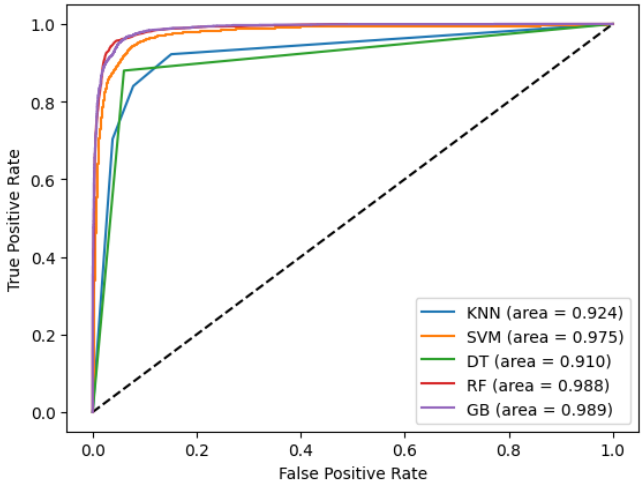


Figura A.17: Comparación de curvas ROC entre modelos superficiales.

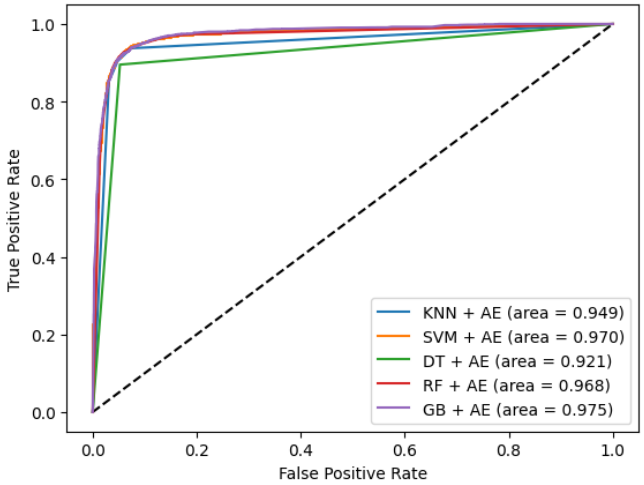
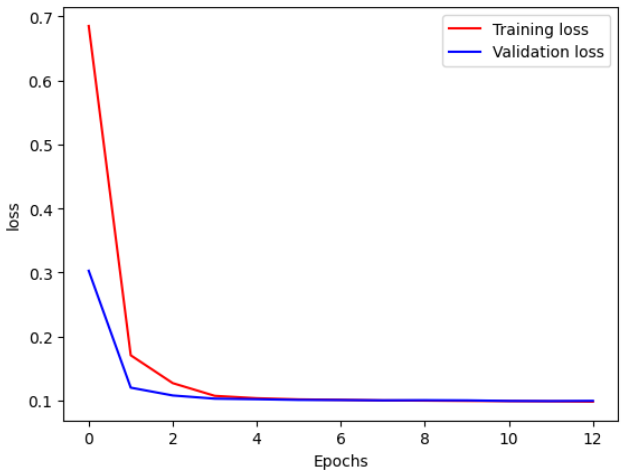


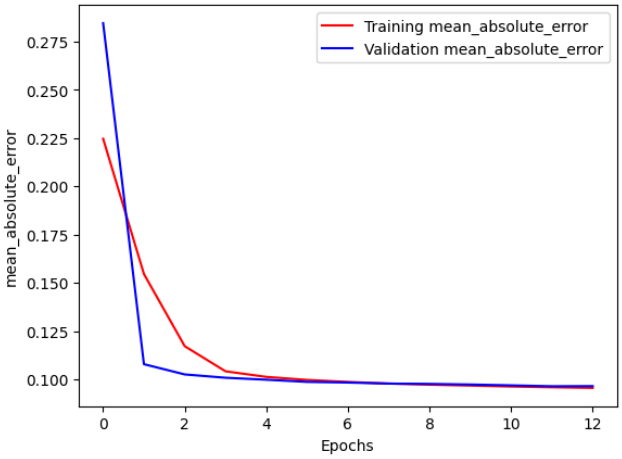
Figura A.18: Comparación de curvas ROC entre modelos superficiales basados en *autoencoder*.

A.3. Resultados predicción sMCI/pMCI

A.3.1. Gráficas generadas en la etapa del entrenamiento y fine tuning del autoencoder

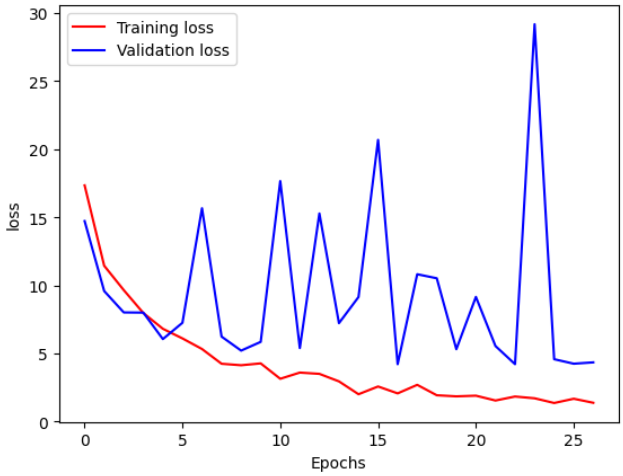


(a) Valores generados por la función de pérdida.

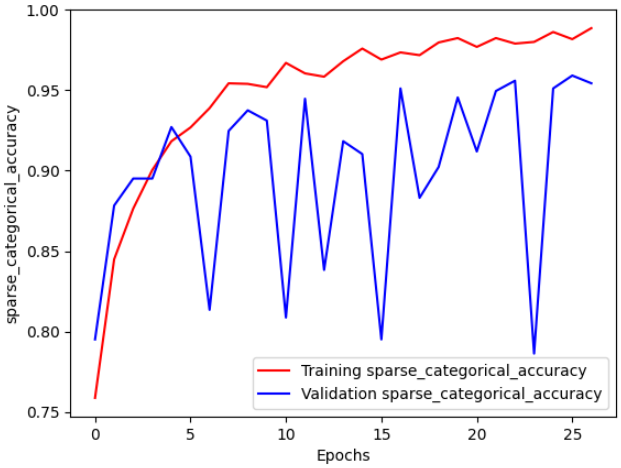


(b) Valores del error absoluto medio generados.

Figura A.19: Gráficas de métricas de función de pérdida y MAE generadas durante el entrenamiento del Autoencoder.



(a) Valores generados por la función de pérdida.



(b) Valores de la métrica *sparse categorical accuracy*.

Figura A.20: Gráficas de métricas de función de pérdida y *sparse categorical accuracy* generadas durante el entrenamiento del Autoencoder en la etapa *fine tuning*.

Matrices de confusión de modelos

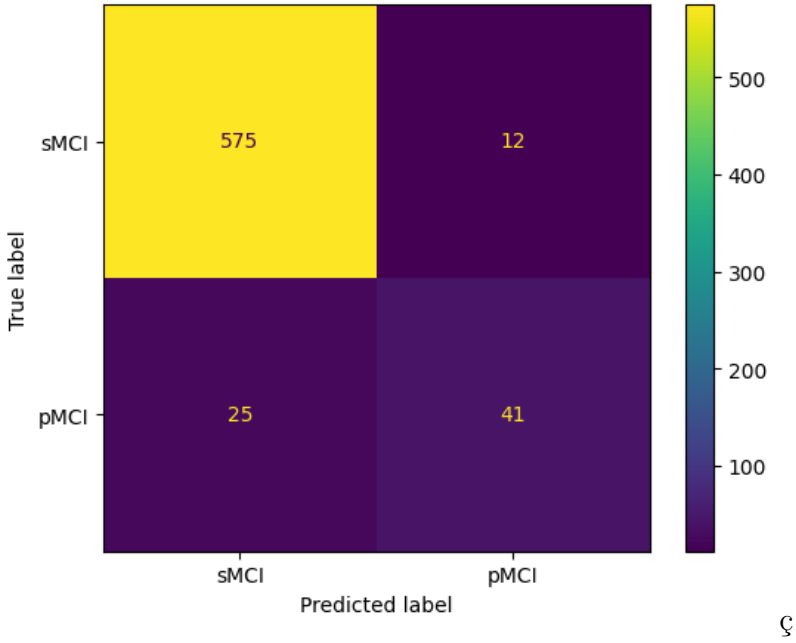
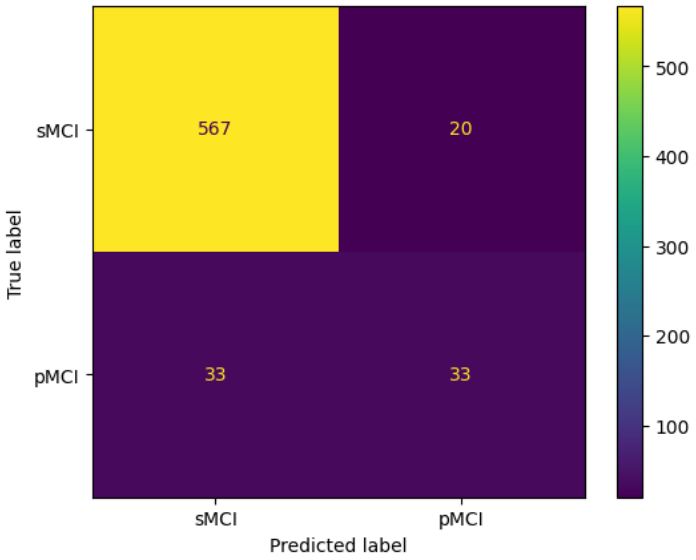
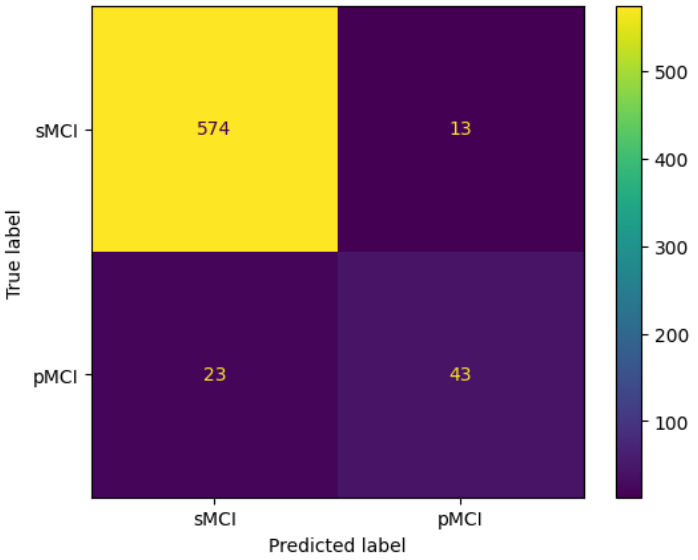


Figura A.21: Matriz de confusión del autoencoder.

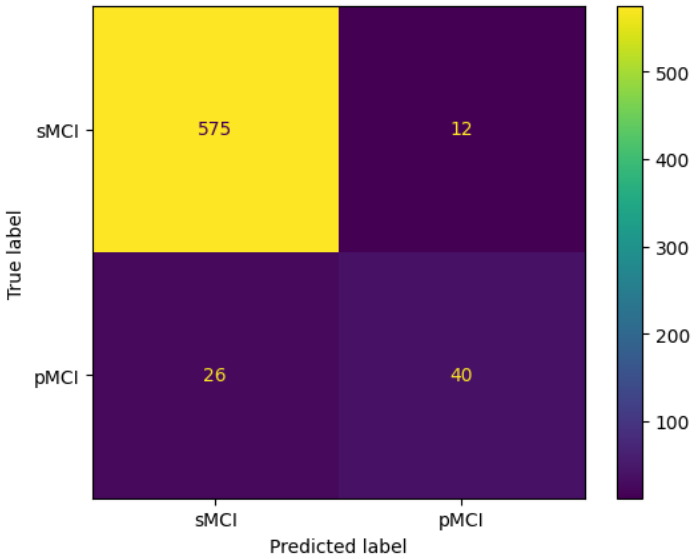


(a) Matriz de confusión modelo kNN.

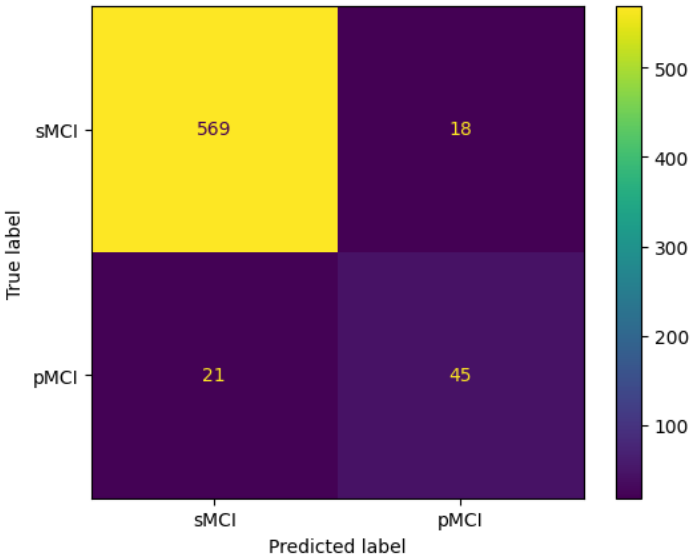


(b) Matriz de confusión modelo kNN + *autoencoder*.

Figura A.22: Matrices de confusión entre kNN y *autoencoder*.

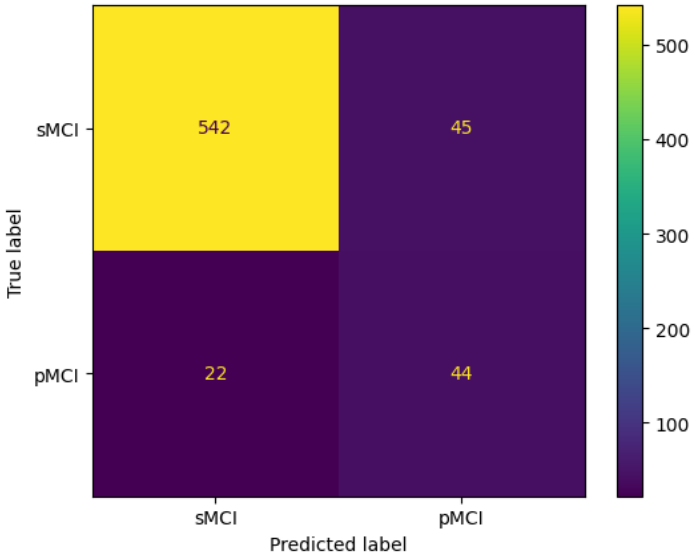


(a) Matriz de confusión modelo SVM.

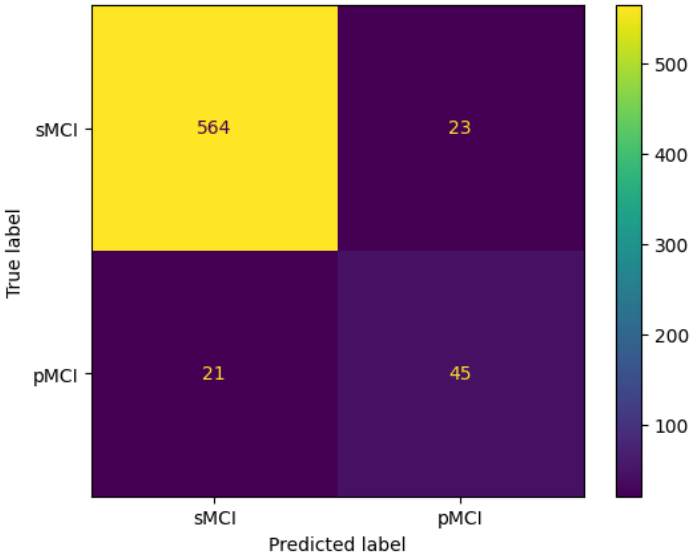


(b) Matriz de confusión modelo SVM + *autoencoder*.

Figura A.23: Matrices de confusión entre SVM y *autoencoder*.

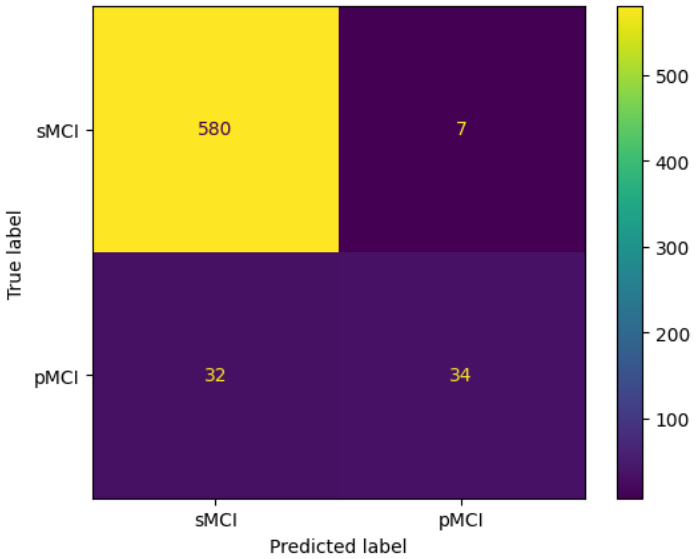


(a) Matriz de confusión modelo DT.

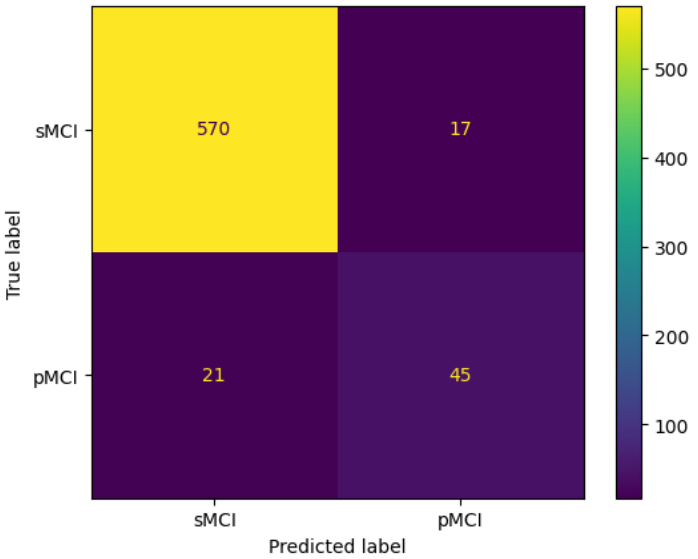


(b) Matriz de confusión modelo DT + *autoencoder*.

Figura A.24: Matrices de confusión entre DT y *autoencoder*.

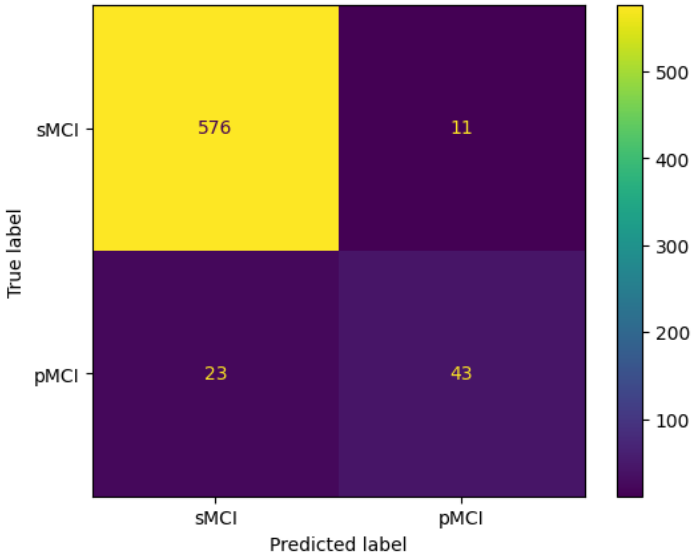


(a) Matriz de confusión modelo RF.

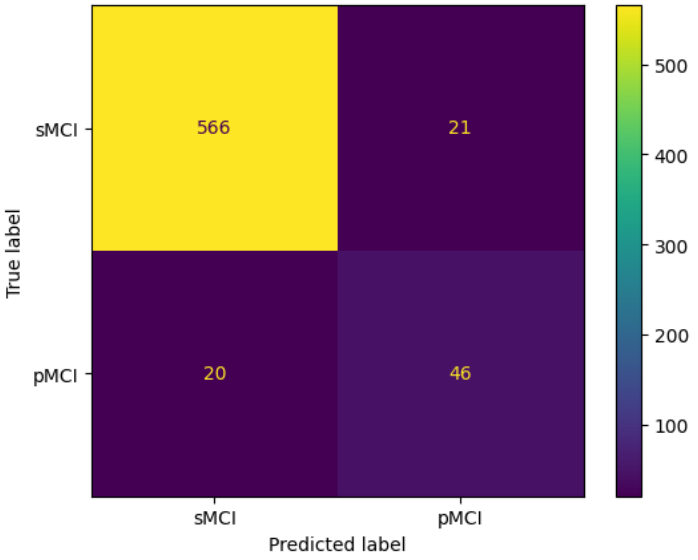


(b) Matriz de confusión modelo RF + *autoencoder*.

Figura A.25: Matrices de confusión entre RF y *autoencoder*.



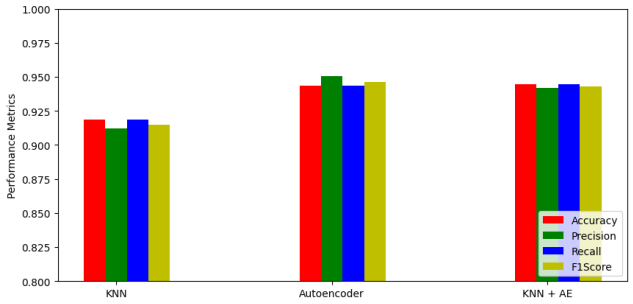
(a) Matriz de confusión modelo GB.



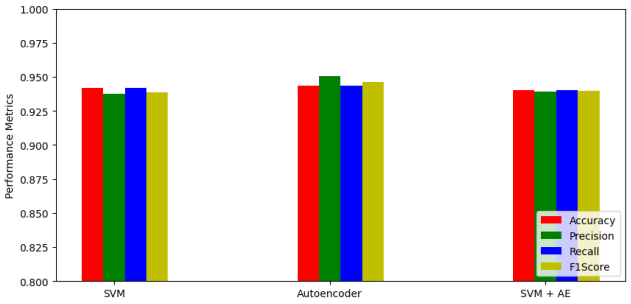
(b) Matriz de confusión modelo GB + *autoencoder*.

Figura A.26: Matrices de confusión entre GB y *autoencoder*.

A.3.2. Métricas de evaluación entre modelos

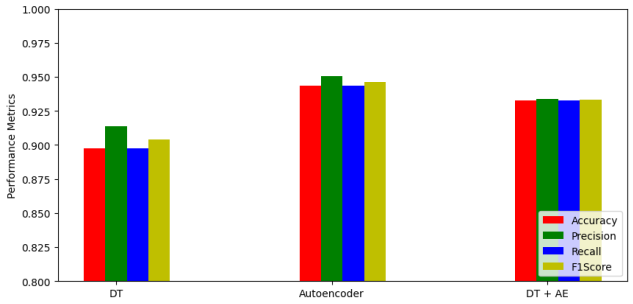


(a) Comparación de métricas de modelo kNN.

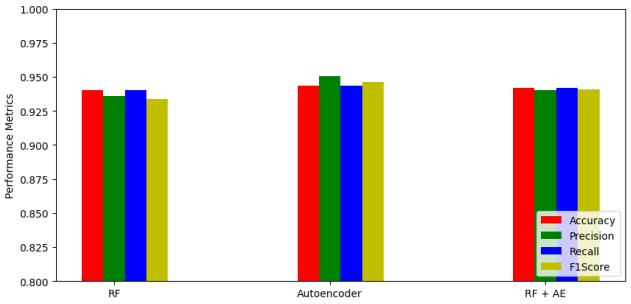


(b) Comparación de métricas de modelo SVM.

Figura A.27: Métricas de modelo kNN y SVM.



(a) Comparación de métricas de modelo DT.



(b) Comparación de métricas de modelo RF.

Figura A.28: Métricas de modelo DT y RF.

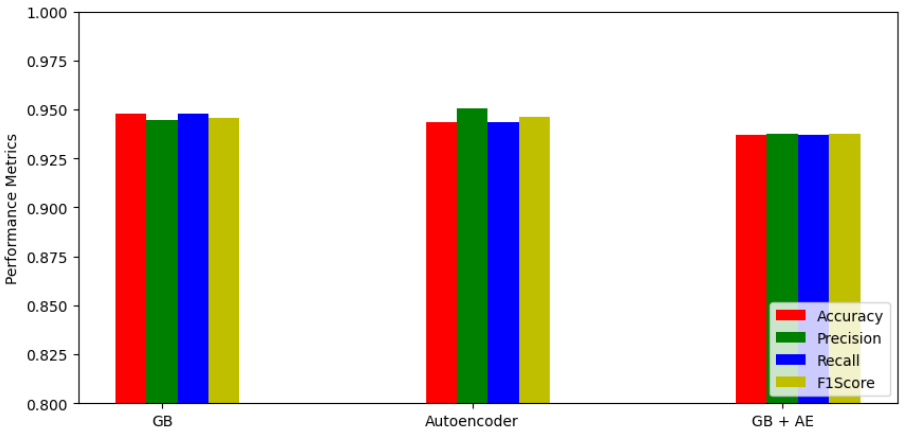


Figura A.29: Comparación de métricas de modelo GB.

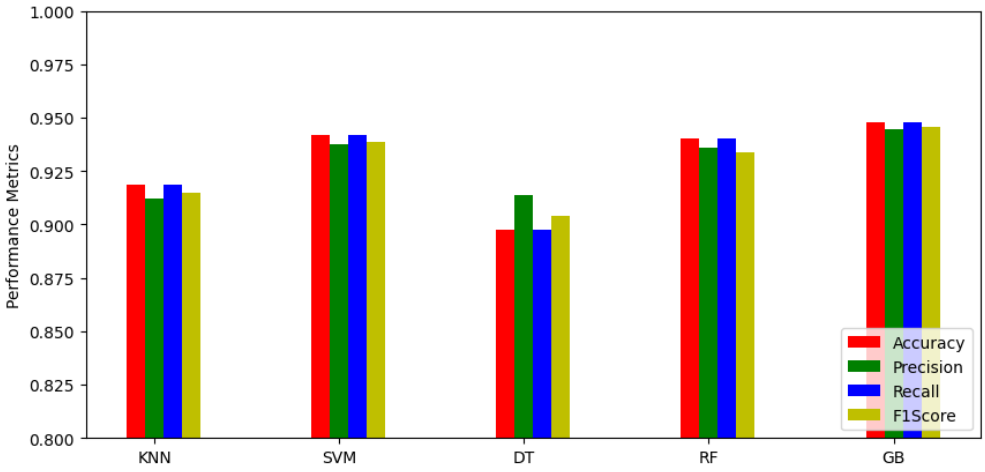


Figura A.30: Comparación de métricas entre modelos superficiales.

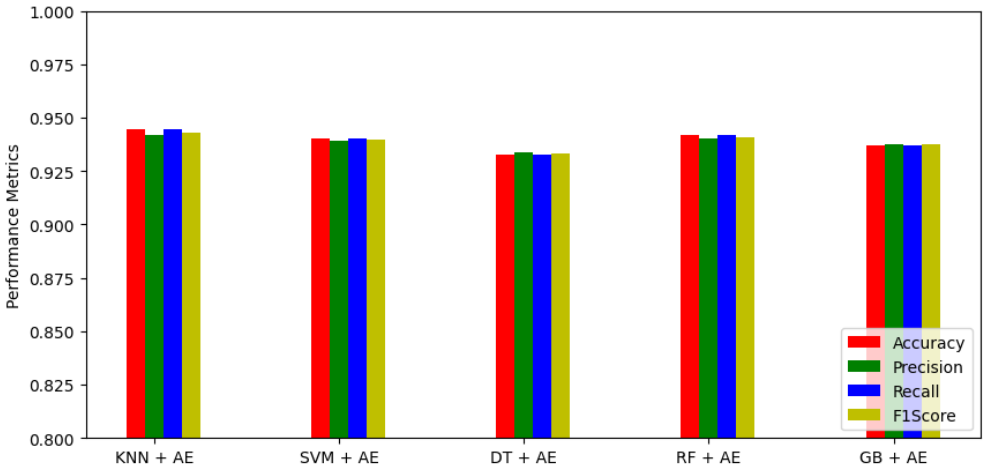
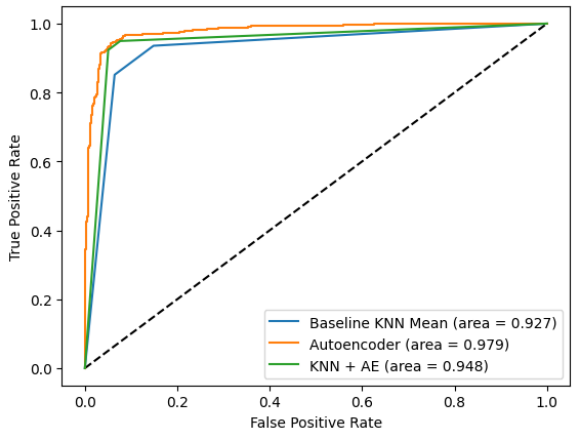
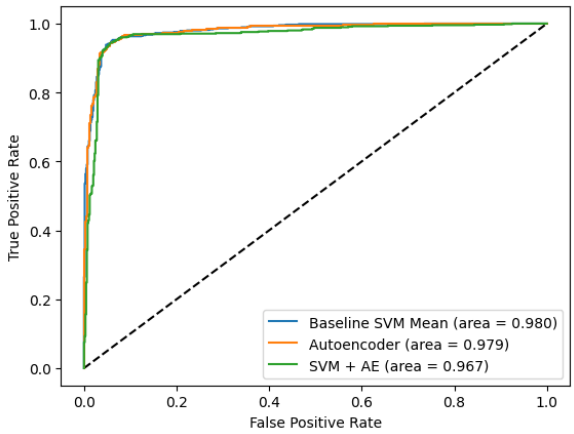


Figura A.31: Comparación de métricas entre modelos superficiales basados en *autoencoder*.

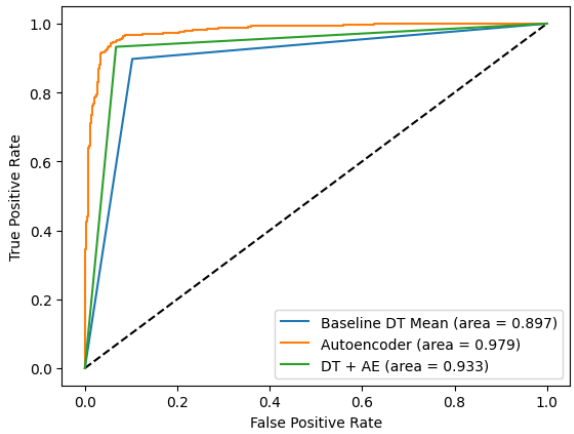
A.3.3. Curvas ROC entre modelos

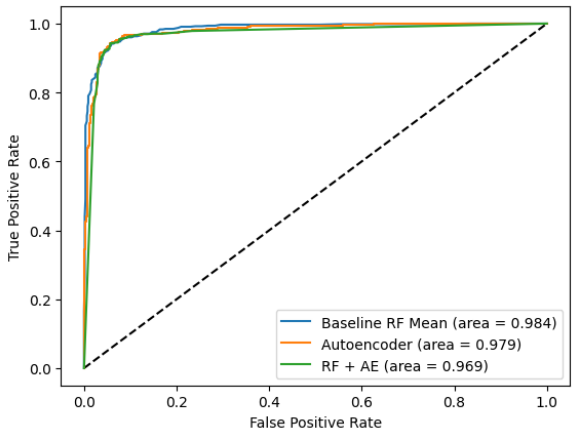




(a) Comparación de curvas ROC de modelo kNN. (b) Comparación de curvas ROC de modelo SVM.

Figura A.32: Métricas de modelo kNN y SVM.





(a) Comparación de curvas ROC de modelo DT. (b) Comparación de curvas ROC de modelo RF.

Figura A.33: Métricas de modelo DT y RF.

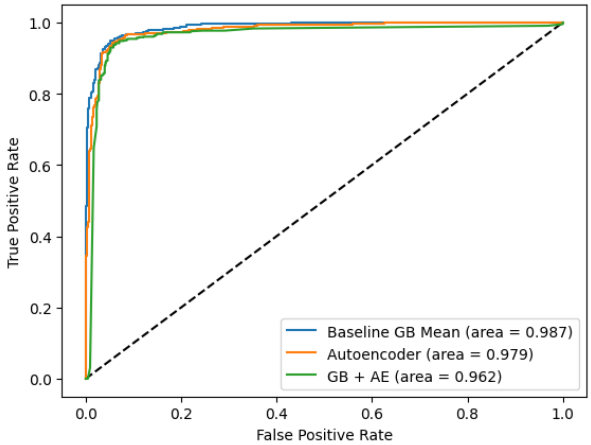


Figura A.34: Comparación de curvas ROC de modelo GB.

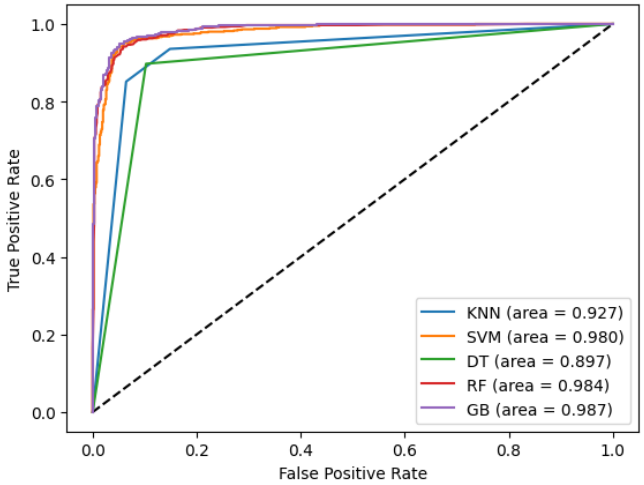


Figura A.35: Comparación de curvas ROC entre modelos superficiales.

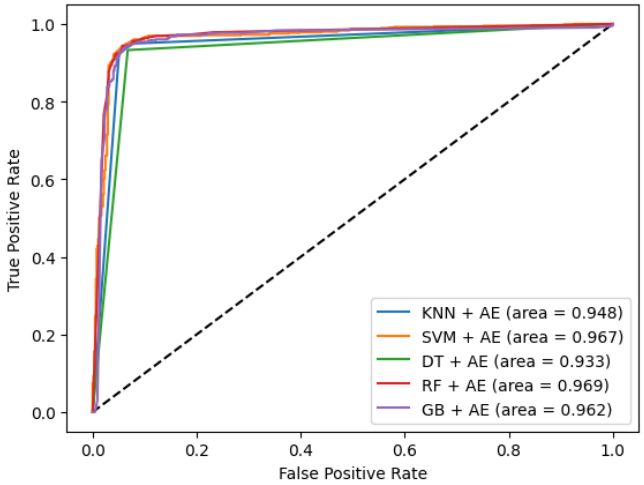


Figura A.36: Comparación de curvas ROC entre modelos superficiales basados en *autoencoder*.

Anexos B

Diagrama de Gantt y tareas realizadas

En las primeras etapas del trabajo se me pidió la realización de un resumen del artículo de *Shaker El Sappagh et al "A multilayer multimodal detection and prediction model based on explainable artificial intelligence for Alzheimer's disease*" [25] para ver si era posible replicarlo, en el resumen se nombraban los métodos utilizados y el lenguaje de programación en donde se puede implementar, también se indica el tipo de datos que se ha utilizado y donde se pueden obtener, además de los pasos que ha realizado en el procesamiento de estos. Una vez entregado el resumen, las profesoras me indicaron una serie de cursos sobre *deep learning* para repasar conceptos y comencé a realizar el proyecto, en primer lugar se empezó con el procesamiento de los datos, una versión simple de autoencoder para poder generar resultados y una primera versión de la memoria. Tras las reunión, se dijo de añadir una comparación entre los datos del trabajo de *El Sappagh et al* [25] y los datos de *TADPOLE challenge* usando un modelo *random forest* para comprobar que se puede obtener un modelo preciso, también se me pidió añadir la predicción sMCI/pMCI al trabajo, en esta etapa encontré varios errores tanto en la parte de procesado de los datos como en los *tied weights* del *autoencoder*, también programé una versión propia de *grid search* mediante un anidamiento de bucles y el uso de *k-fold cross validation*, también hice comparaciones de tiempo de ejecución y métricas entre los modelos, al igual que la generación de resultados mediante la librería *matplotlib* para añadirlos a la memoria, en el proceso, optimicé el código, arreglé errores y comenté las funciones que tenía. Tras entregar una versión mejorada de la memoria, se me pidió cambiar varias secciones y corregir errores de redacción, en esta última etapa añadí el coeficiente *Kappa* de *Cohen* a la evaluación de modelos e investigué sobre las etapas del diagnóstico del *Alzheimer*. Durante los últimos meses del curso estuve realizando cambios a la memoria según las indicaciones de mis profesoras para

que fuera más presentable y bien redactada.

En la imagen de la Figura B.1 se puede ver todas las tareas realizadas y el periodo de tiempo en el que se han hecho.

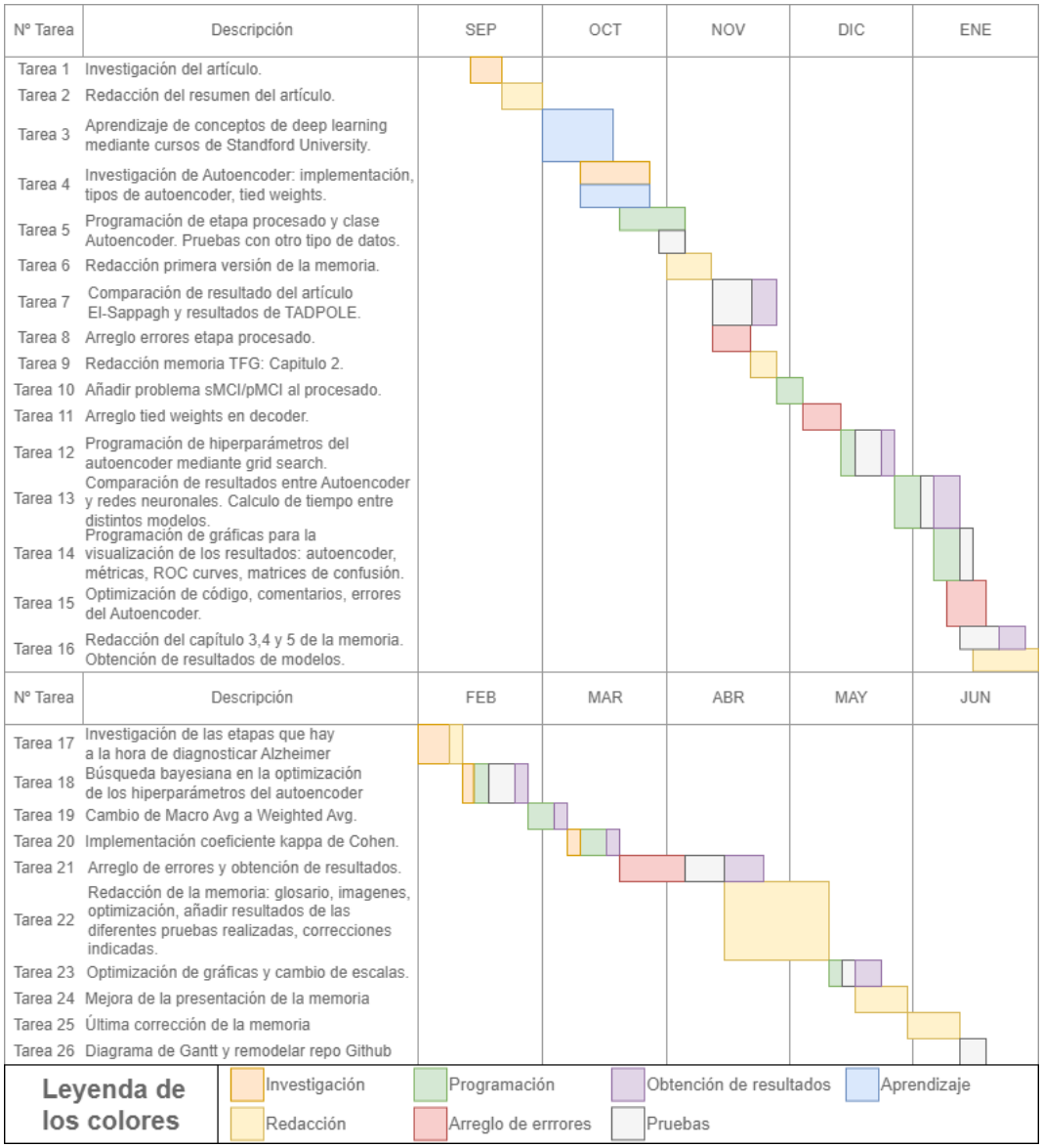


Figura B.1: Diagrama de Gantt del proyecto.

Anexos C

GITHUB

El código se encuentra en el siguiente enlace: `https://github.com/aFacorroLoscos/SIAEPADDTB`.

Dentro del repositorio están indicadas las librerías que se han utilizado para este proyecto al igual que un resumen de cómo funciona el programa. Por último, se indica que lineas de comando hay que ejecutar para generar un procesado de los datos o una evaluación de los modelos, también se explica el significado de cada uno de los *flags*.